REALSYSTEM IQ PRODUCTION GUIDE

RealONE Update

Last Update: 21 September 2001

RealNetworks, Inc.
PO Box 91123
Seattle, WA 98111-9223
U.S.A.

http://www.real.com
http://www.realnetworks.com

©2001 RealNetworks, Inc. All rights reserved.

Information in this document is subject to change without notice. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of RealNetworks, Inc.

Printed in the United States of America.

Annabelle, Film.com, The Future is Real, Goldpass, Musicnet, Neuralcast, Progressive Networks, RBN, the Real "bubble" (logo), Real Broadcast Network, RealArcade, RealAudio, Real.com, RealDownload, RealJukebox, RealMedia, RealMP, RealNetworks, RealPlayer, RealONE Player, RealPresenter, RealProducer, RealProducer Plus, RealProducer Pro, RealProxy, RealPublisher, RealSites, RealSlideshow, RealStore.com, RealSystem, RealText, RealVideo, SureDownload, SureStream, Surreal.FX Design, Take5, and WebActive are trademarks or registered trademarks of RealNetworks, Inc.

Other product and corporate names may be trademarks or registered trademarks of their respective companies.

SUMMARY OF CONTENTS

DOCUMENTATION RELEASE NOTE *1*

INTRODUCTION *3*

PART I: GETTING STARTED WITH STREAMING MEDIA

1 NEW FEATURES *15*

2 PRESENTATION PLANNING *23*

PART II: PRODUCING REALSYSTEM CLIPS

3 AUDIO PRODUCTION *43*

4 VIDEO PRODUCTION *57*

5 FLASH ANIMATION *75*

6 REALTEXT MARKUP *91*

PART III: LEARNING SMIL

7 SMIL BASICS *127*

8 CLIP SOURCE TAGS *145*

PART IV: ORGANIZING A PRESENTATION

9 PRESENTATION INFORMATION *165*

10 GROUPS *173*

11 LAYOUT *195*

PART V: TIMING AND LINKING CLIPS

12 BASIC TIMING *237*

13 ADVANCED TIMING *261*

14 HYPERLINKS *279*

PART VI: MASTERING ADVANCED SMIL FEATURES

15 TRANSITION EFFECTS *315*

16 ANIMATIONS *341*

17 SWITCHING *363*

18 PREFETCHING *383*

PART VII: STREAMING YOUR CLIPS

19 WEB PAGE EMBEDDING *395*

20 PRESENTATION DELIVERY *415*

PART VIII: APPENDIXES

A BASIC QUESTIONS *435*

B PRODUCTION TASKS *445*

C SMIL TAG REFERENCE *453*

D SMIL COLOR VALUES *475*

E REALTEXT TAG REFERENCE *479*

F FILE TYPE REFERENCE *485*

G LANGUAGE CODES *487*

GLOSSARY *489*

INDEX *497*

CONTENTS

DOCUMENTATION RELEASE NOTE *1*

INTRODUCTION *3*

What This Guide Covers 3
How to Download This Guide to Your Computer 5
How this Guide Is Organized 5
Conventions Used in this Guide 9
Additional RealSystem Resources 10
Technical Support 10

PART I: GETTING STARTED WITH STREAMING MEDIA

1 NEW FEATURES *15*

RealONE Player Introduced 15
SMIL 2.0 Support 15
New Clip Tag Attributes 16
Expanded Grouping Possibilities 17
Enhanced Layout Choices 18
More Timing Possibilities 19
New Linking Attributes 19
Clip Transition Effects 20
SMIL Animations 20
Powerful Content Control Capabilities 21
Additions and Deletions to this Guide 21

2 PRESENTATION PLANNING *23*

Step 1: Decide How to Deliver Clips 23
RealServer Streaming 23
Web Server Downloading 24
Local Playback 25
Step 2: Choose Clip Types and Gather Tools 25
Audio and Video 25
SMIL 27
Animation 27
Images 28

Text 29
Autoupdate Feature 29
Step 3: Develop a Bandwidth Strategy 31
Buffering 31
Audience Bandwidth Targets 32
Clip Bandwidth Characteristics 33
Reaching Multiple Audiences 35
Step 4: Organize the Presentation Timeline 37
Timeline Considerations 37
Timelines for Multiclip Presentations 38
Timeline Management 39
Step 5: Create Your Clips 40

PART II: PRODUCING REALSYSTEM CLIPS

3 AUDIO PRODUCTION *43*

Understanding RealAudio 43
Bandwidth and Audio Quality 43
RealAudio Bandwidth Characteristics 44
RealAudio Codecs 45
Steps for Streaming RealAudio 49
Capturing Audio 50
Source Media 51
Recording Equipment 51
Shielded Cables 51
Input Levels 51
Volume Levels for Live Broadcasts 52
Sampling Rates 52
Optimizing Audio 52
DC Offset 53
Normalization 53
Dynamics Compression 53
Equalization 54
Encoding RealAudio with RealProducer 54
RealAudio in Multiclip Presentations 55
Audio Compatibility with RealPlayer 5 55

4 VIDEO PRODUCTION *57*

Understanding RealVideo 57
RealVideo Bandwidth Characteristics 57
RealVideo Frame Rates 59
RealVideo Clarity 60
RealVideo Dimensions 61

RealVideo Codecs....63
Scalable Video Technology....64
Steps for Streaming RealVideo....64
Recording Video....66
Source Media Quality....66
Video Staging....66
Scene Changes and Movement....67
Colors and Lighting....67
Digitizing Video....67
S-Video Output....68
Color Depth....68
Digitized Video Formats....68
Video Capture Frame Rates....68
Video Capture Screen Sizes....68
Computer Speed and Disk Space....69
Video Source File Size Limit....69
Encoding RealVideo with RealProducer....70
RealVideo Streaming Speeds....71
Video Compatibility with RealPlayer 5....71
RealVideo Filters....72
RealVideo Options....73

5 FLASH ANIMATION *75*

Understanding Flash....75
RealSystem Software Versions for Flash....76
Flash Bandwidth Characteristics....76
Flash Clip Size....77
Flash CPU Use....78
Adding Audio to Flash....79
Bandwidth Division Between Flash and RealAudio....80
Tips for Choosing RealAudio Codecs....82
Using Interactive Flash Commands....82
Flash Clip Timeline Commands....82
RealONE Player Commands....83
Go To Commands....84
Load Movie Commands....85
Secure Transactions....87
Mouse Events....87
Streaming a Flash Clip....88

6 REALTEXT MARKUP *91*

Understanding RealText....91
RealText Language Support....92

Structure of a RealText Clip 92
Rules for RealText Markup 93
RealText Bandwidth 93
RealText in a SMIL Presentation 93
RealText Broadcast Application 94
Setting RealText Window Attributes 94
Specifying the Window Type 94
Setting the Window Size and Color 96
Setting the Clip Duration 97
Adding a Version Number 99
Specifying Hyperlink Appearance 100
Controlling Text Flow 100
Timing and Positioning Text 102
Controlling When Text Appears and Disappears 102
Clearing Text from the Window 104
Positioning Text in a Window 105
Aligning Text in a Tickertape Window 105
Ensuring Text Delivery 106
Specifying Languages, Fonts, and Text Colors 106
Specifying the Character Set 106
Setting the Font 109
Setting the Text Size 111
Controlling Text Colors 112
Controlling Text Layout and Appearance 113
Adding Space Between Text Blocks 113
Centering Text 114
Preformatting Text 114
Using HTML-Compatible Tags 115
Emphasizing Text 115
Creating Links and Issuing Commands 116
Creating a Mail Link 116
Opening a URL in RealONE Player or a Browser 116
Issuing RealONE Player Commands 117
Using Coded Characters 118
Using Coded Characters with the mac-roman Character Set 119
RealText Examples 119
Generic Window 119
Tickertape Window 120
Scrolling News Window 121
Teleprompter Window 122

PART III: LEARNING SMIL

7 SMIL BASICS *127*

Understanding SMIL....127
Advantages of Using SMIL....127
SMIL 1.0 and SMIL 2.0....129
SMIL 2.0 Modules....129
SMIL 2.0 Profiles....131
Interoperability Between SMIL-Based Players....132
Creating a SMIL File....133
The SMIL 2.0 Tag and Namespace....134
Header and Body Sections....134
Tags, Attributes, and Values....135
Binary and Unary Tags....137
SMIL Recommendations....137
SMIL Tag ID Values....138
Using Customized SMIL Attributes....139
RealNetworks Extensions Namespace....140
A Closer Look at Namespaces....140
Tips for Defining Namespaces....141
Viewing SMIL Source Markup....141
Playback Differences from SMIL 1.0....142
Behavioral Changes....142
Updating SMIL 1.0 Files to SMIL 2.0....143

8 CLIP SOURCE TAGS *145*

Creating Clip Source Tags....145
Adding a Clip ID....146
Setting a Clip's Streaming Speed....146
Creating a Brush Object....148
Using a Ram File as a Source....149
Using a SMIL File as a Source....150
Writing Clip Source URLs....151
Linking to Local Clips....151
Creating a Base URL....152
Linking to Clips on RealServer....153
Linking to Clips on a Web Server....153
Caching Clips on RealONE Player....154
Modifying Clip Colors....157
Adjusting Clip Transparency and Opacity....158
Substituting Transparency for a Specific Color....159
Substituting a Color for Transparency....162

PART IV: ORGANIZING A PRESENTATION

9 PRESENTATION INFORMATION *165*

Understanding Presentation Information 165
Information Encoded in Clips 165
Clip Source Tag and Group Information 166
SMIL Presentation Information 166
Accessibility Information 167
RealONE Player Context Window 167
Coded Characters 167
Adding Clip and Group Information 168
Tips for Defining Clip Information 168
Defining Information for the SMIL Presentation 169
Managing Presentation Information 169
Adding Accessibility Information 169
Including an Alternate Clip Description 170
Using a Long Description 170
Setting the Clip Read Order 171

10 GROUPS *173*

Understanding Groups 173
Groups Within Groups 174
Playing Clips in Sequence 175
Creating Sequences Without <seq> Tags 176
Tips for Creating Sequences 176
Playing Clips in Parallel 177
Tips for Creating Parallel Groups 177
Synchronizing Playback in Parallel Groups 178
Creating an Independent Timeline 179
Setting the Synchronization Behavior 179
Specifying Synchronization Behavior Default Values 182
Loosening the Synchronization for Locked Elements 185
Specifying Synchronization Tolerance Default Values 185
Tips for Synchronizing Clips 186
Creating an Exclusive Group 186
Defining Interactive Begin Times 187
Using Clip Interruption 188
Modifying Clip Interruption Behavior 188
Tips for Defining Exclusive Groups and Priority Classes 192

11 LAYOUT *195*

Understanding Layouts 195
Root-Layout Area 195

Playback Regions....196
Subregions....196
Secondary Media Windows....197
Clip Position and Fit....198
Tips for Laying Out Presentations....199
Layout Tag Summary....202
Creating Main and Secondary Media Windows....203
Defining the Main Media Window....203
Creating Secondary Media Windows....204
Changing Resize Behavior....206
Defining Playback Regions....206
Setting Region IDs and Names....207
Defining Region Sizes and Positions....208
Assigning Clips to Regions....213
Stacking Regions That Overlap....214
Adding Background Colors....215
Controlling Audio Volume in a Region....218
Defining Subregions....218
Creating Registration Points....221
Using Alignment Values....221
Defining Registration Points in Clip Source Tags....222
Creating a Reusable Registration Point....224
Defining How Clips Fit Regions....227
fit Attribute Values....227
Overriding a Region's fit Attribute....229
Tips for Defining the fit Attribute....229
Layout Examples....229
Centering a Video on a Background....230
Displaying a Letterbox Clip....230
Turning Down an Audio Clip's Volume....231
Playing Three Clips Side-by-Side....231
Placing a Clip in a Secondary Media Window....232
Playing the Same Clip in Multiple Regions....233

PART V: TIMING AND LINKING CLIPS

12 BASIC TIMING *237*

Understanding Basic Timing....237
Groups Create the Timing Superstructure....237
Timing is Relative to Groups....238
Timing Attributes Covered in this Chapter....238
Specifying Time Values....239

Using Shorthand Time Values 239
Using the Normal Play Time Format 240
Setting Begin and End Times 240
Using a Begin Time with a Clip 240
Using an End Time with a Clip 241
Using Begin and End Times with Groups 241
Setting Internal Clip Begin and End Times 242
Combining clipBegin and clipEnd with begin and end 242
Setting Durations 243
Choosing end or dur 243
Setting a Duration for the Length of Media Playback 244
Using an Indefinite Duration 244
Tips for Setting Durations 245
Ending a Group on a Specific Clip 245
Stopping the Group After the Last Clip Finishes 246
Stopping the Group After the First Clip Finishes 246
Stopping the Group When a Specific Clip Finishes 246
Tips for Using endsync Attributes 247
Repeating an Element 247
Repeating an Element a Certain Number of Times 247
Repeating an Element a Specific Amount of Time 247
Specifying the Length of Each Repeating Cycle 248
Setting a Total Playback Time 248
Looping Playback Indefinitely 248
Stopping a Clip's Encoded Repetitions 249
Managing Bandwidth with Repeating Clips 250
Tips for Repeating Elements 251
Setting a Fill 251
Using an Automatic Fill 252
Setting a Fill with Sequential Clips 253
Setting a Fill in Parallel Groups 254
Setting a Fill in Exclusive Groups 254
Displaying a Clip Throughout a Presentation 255
Summary of Common Clip fill Values 255
Setting a Group Fill 256
Tips for Setting a Fill 257
Specifying a Default Fill 258
Adding a Default Fill to a Group 258
Inheriting a Default Fill from a Containing Group 259

13 ADVANCED TIMING *261*

Understanding Advanced Timing 261

Advanced Timing Syntax 261
Event Types 262
Positive Offset Times 263
Negative Offset Times 265
Multiple Timing Values 266
Defining an Element Start or Stop Event 266
Sample Values 267
Example 267
Defining a Repeat Event 268
Sample Values 269
Example 269
Defining a Mouse Event 269
Sample Values 270
Example 271
Defining a Keyboard Event 271
Sample Values 272
Example 272
Tips for Defining Keyboard Events 272
Defining a Secondary Window Event 273
Sample Values 274
Example 274
Using Media Markers 274
Coordinating Clips to an External Clock 274
Setting Minimum and Maximum Times 274
Controlling Whether an Element Restarts 274
Setting a Default Restart Value 275

14 HYPERLINKS *279*

Understanding Hyperlinks 279
Links to HTML Pages 279
Links to Streaming Media 281
Methods of Activating a Link 282
General Tips for Creating Hypertext Tags 282
Creating a Simple Link 283
Using the <area/> Tag 283
Creating a Timed Link 284
Defining Hot Spots 285
Defining Basic Hyperlink Properties 290
Specifying the Link URL 290
Leaving Out a URL Reference for Hot Spots 291
Opening a Link on a Keystroke 291
Opening a URL Automatically 293

Displaying Alternate Link Text 293
Setting a Tab Index for Multiple Links 294
Linking to HTML Pages 294
Selecting a Browsing Window 295
Opening HTML Pages in the Context Window 297
Targeting a Frame or Named Window 299
Controlling the Media Playback State 300
Tips for Opening HTML Page Links 300
Linking to Streaming Media 301
Replacing the Source Presentation 301
Opening a New Media Window with SMIL 302
Linking to a SMIL Fragment 304
Adjusting Audio Volumes in Linked Presentations 306
Opening a New Media Window Through RealText or Flash 306
Hyperlink Examples 309
Displaying a Web Page when a Presentation Ends 310
Opening Web Pages During a Presentation 310

PART VI: MASTERING ADVANCED SMIL FEATURES

15 TRANSITION EFFECTS *315*

Understanding Transition Effects 315
Timelines and Transition Effects 316
Layouts and Transition Effects 316
Animations and Transition Effects 316
Audio and Transition Effects 316
Multiple Clips with Transition Effects 317
Summary of Transition Effects Tags 317
Defining Transition Types 317
Edge Wipe Transition Effects 318
Iris Wipe Transition Effects 321
Clock Wipe Transition Effects 323
Matrix Wipe Transition Effects 326
Fade, Push, and Slide Transition Effects 329
Modifying Transition Effects 330
Setting a Transition Effect's Duration 331
Reversing a Transition Effect's Direction 331
Using Partial Transition Effects 332
Repeating Transition Effects Horizontally or Vertically 333
Setting a Border Width 334
Defining Colors and Border Blends 334
Assigning Transition Effects to Clips 335

Using Clip Fills with Transition Effects 336
Transition Effects Examples 338
Fading to a Color Between Clips 338
Crossfading Videos 339

16 ANIMATIONS *341*

Understanding Animations 341
Animation Tags 342
Animation Tag Placement 342
SMIL Timing with Animations 344
Simultaneous Animations 345
Creating Basic Animations 345
Selecting the Element and Attribute to Animate 346
Defining Simple Animation Values 350
Defining a Range of Animation Values 352
Controlling How an Animation Flows 353
Jumping from Value to Value 353
Moving Linearly from Point to Point 354
Flowing at an Even Pace 354
Creating Additive and Cumulative Animations 355
Adding Animation Values to a Base Value 355
Making Animations Repeat and Grow 356
Using the Specialized Animation Tags 358
Animating Colors 358
Creating Horizontal and Vertical Motion 359
Setting an Attribute Value 360
Manipulating Animation Timing 360

17 SWITCHING *363*

Understanding Switching 363
Creating a Switch Group 363
Adding a Default Option to a Switch Group 364
Using Inline Switching 365
Available Test Attributes 366
Tips for Writing Switch Groups 366
Switching Between Language Choices 367
Setting Language Codes 368
Providing Subtitles or Overdubbing 369
Switching Between Bandwidth Choices 370
Switching with SureStream Clips 371
Enhancing Presentation Accessibility 372
Switching Based on the Viewer's Computer 373
Switching for CPU Type 373

Switching for Operating System 374
Switching for Monitor Size or Color Depth 375
Switch Group Examples 377
Switching with Multiple Attributes 377
Switching for Different Video Sizes 378
Displaying System Captions Using RealText 379

18 PREFETCHING *383*

Understanding Prefetching 383
Using the <prefetch/> Tag 384
Managing Prefetch Bandwidth 385
Specifying Prefetch Bandwidth in Bits Per Second 385
Specifying Prefetch Bandwidth as a Percentage 386
Controlling Prefetch Data Download Size 387
Prefetching a Specific Amount of Data 387
Prefetching a Specific Length of a Clip's Timeline 388
Tips for Prefetching Data 388
RealAudio and RealVideo Prefetching 388
Prefetch URLs 389
SMIL Timing with Prefetching 389
Prefetch Testing 390
Prefetching Examples 390
Displaying an Image Until Prefetching Completes 390
Prefetching and Caching an Image 391

PART VII: STREAMING YOUR CLIPS

19 WEB PAGE EMBEDDING *395*

Understanding Web Page Embedding 395
<EMBED> Tags 395
Layout Possibilities 396
RealONE Player Controls 397
Javascript and ActiveX 397
Using <EMBED> Tags 398
Specifying the Source 398
Setting the Width and Height 400
Turning off the Java Virtual Machine 400
Supporting Other Browsers 401
Adding RealONE Player Controls 401
Basic Controls 402
Individual Controls and Sliders 403
Information Panels 405
Status Panels 406

Linking Multiple Controls 407
Controlling Image Display 409
Setting a Background Color 409
Centering a Clip 410
Maintaining a Clip's Aspect Ratio 410
Suppressing the Real Logo 410
Setting Automatic Playback 411
Starting a Presentation Automatically 411
Looping a Presentation Continuously 411
Specifying a Number of Loops 412
Setting Shuffle Play 412
Laying Out SMIL Presentations 412
Defining the Layout with SMIL 412
Defining the Layout with HTML 413

20 PRESENTATION DELIVERY *415*

Understanding Linking and URLs 415
The Ram File 415
The Difference Between RTSP and HTTP 417
Web Server MIME Types 418
Placing Clips on a Server 419
Understanding Directory Paths and URLs 419
Copying Clips to a Server 419
Launching RealONE Player with a Ram File 420
Writing a Ram File 420
Setting a Presentation's Starting Mode 422
Adding Comments to a Ram File 423
Examples of Linking a Web Page to Clips 423
Using Ramgen for Clips on RealServer 425
Linking Your Web Page to RealServer Using Ramgen 426
Listing Alternative Presentations with Ramgen 427
Combining Ramgen Options 427
Limitations on Web Server Playback 428
No SureStream Clips Encoded for Multiple Bandwidths 428
No Secure RealAudio and RealVideo Clips 428
Limited Ability to Keep Parallel Clips Synchronized 428
No Way to Set Image Streaming Speeds 429
RealPix Presentations Require Clip Size Information 429
SMIL File Optional 429
SMIL Internal Timing Commands Do Not Work 429
No Presentation Seeking 430
No RTSP URLs 430

No Live Broadcasting 430
Testing Your Presentation 430
Advertising on RealGuide 431
Using RealNetworks Logos 432

PART VIII: APPENDIXES

A BASIC QUESTIONS *435*

Creating Streaming Clips 435
Getting Production Tools 436
Using SureStream 438
Writing SMIL Files 438
Streaming Clips 439
Advertising 441
Broadcasting 442
Getting Technical Support 444

B PRODUCTION TASKS *445*

Basic SMIL Issues 445
Clips and URLs 446
Colors and Transparency 446
Layouts 447
Basic Timing and Groups 448
Advanced Timing 448
Hyperlinks 449
Special Effects 449
Advanced Streaming 450
Web Page Embedding 450

C SMIL TAG REFERENCE *453*

<smil>...</smil> 453
Header Tags 453
<meta/> 453
<layout>...</layout> 454
<transition/> 458
Clip Source Tags 458
<prefetch/> 461
Group Tags 462
<seq>...</seq> 462
<par>...</par> 463
<excl>...</excl> 464
<switch>...</switch> 466
Hyperlink Tags 468

<a>... 468
<area/> 468
Animation Tags 470
<animate/> 470
<animateColor/> 471
<animateMotion/> 472
<set/> 474

D SMIL COLOR VALUES *475*
Specifying RGB Color Values 475
Using Standard RGB Color Values 475
Specifying RGB Percentages 476
Defining Hexadecimal Color Values 476
Using Six-Digit Hexadecimal Values 476
Defining a Three-Digit Hexadecimal Value 477
Using Color Names 477
Tips for Defining Color Values 478

E REALTEXT TAG REFERENCE *479*
Window Tag Attributes 479
Time and Position Tags 480
Layout and Appearance Tags 481
Font Tag Attributes 482
Hyperlinking Commands 482

F FILE TYPE REFERENCE *485*

G LANGUAGE CODES *487*

GLOSSARY *489*

INDEX *497*

DOCUMENTATION RELEASE NOTE

This guide aims primarily to explain the use of the SMIL 2.0 standard, and the RealNetworks extensions to SMIL 2.0, in RealONE Player. Please note the following about this guide:

- See Chapter 1 for summaries of new SMIL 2.0 features, as well as for information about additional changes to this manual from RealSystem Release 8.
- This guide documents a pre-release product. RealNetworks has attempted to make this guide as thorough and as accurate as possible, but you may find discrepancies between the instructions in this guide and RealONE Player performance. Known issues include the following:
  - Secondary windows, which are described in the section "Secondary Media Windows" on page 197, are currently plain windows that do not include the standard RealONE Player skin.
  - Contrary to the information in "Replacing the Source Presentation" on page 301, a source clip stops, rather than pauses, when another clip replaces it in its media window.
  - Push wipe transitions, described in "Fade, Push, and Slide Transition Effects" on page 329, are not yet functional.
- The following aspects of RealONE Player functionality have not yet been documented:
  - How clip and presentation information displays in the RealONE Player user interface. The section "Managing Presentation Information" on page 169 is a placeholder for this information.
  - Support for media markers. The section "Using Media Markers" on page 274 is a placeholder for this information.
  - Support for wallclock timing. The section "Coordinating Clips to an External Clock" on page 274 is a placeholder for this information.

  - The min and max timing attributes. The section "Setting Minimum and Maximum Times" on page 274 is a placeholder for this information.
  - Time manipulations for animations. The section "Manipulating Animation Timing" on page 360 is a placeholder for this information.
  - The skip-content attribute.
  - The <metadata> tag.
  - The systemComponent and systemRequired attributes for <switch> tags.
- This guide will grow over the next several months to include more information, authoring tips, and samples. Be sure to check for updates frequently.
- Thank you for previewing RealONE Player!

INTRODUCTION

RealSystem™ gives you the power to stream compelling multimedia presentations over a network. It includes RealServer™, the most advanced streaming media server available, along with RealONE™ Player and RealPlayer®, the world's most popular applications for playing streaming media clips. This production guide will help you produce any multimedia presentation, whether it is a simple video on your home page or a multimedia extravaganza.

> **Tip:** To experience the many possibilities of streaming media, download RealONE Player from **http://www.real.com**, and then visit **http://realguide.real.com**.

What This Guide Covers

This production guide tells you how to assemble a RealSystem presentation. Although it provides many tips for producing streaming media, the more you know about producing audio, video, and graphics in general, the faster you will be able to create a great streaming presentation. Topics in this guide fall into four general areas:

- Planning a Presentation

  Before you launch into streaming media production, you need to consider several issues carefully. What is your target bandwidth? What types of clips will you use? How will your presentation timeline progress? Addressing these issues is critical for producing a successful presentation. To learn the basics, start with "Chapter 2: Presentation Planning" beginning on page 23.

- Producing Clips

  RealSystem streams a core set of clip types: RealAudio, RealVideo, Flash, RealText, and RealPix. You can stream just a single clip, or combine

various clips into a complex presentation. "Part II: Producing RealSystem Clips" beginning on page 41 explains these clip types.

> **Note:** This guide does not explain how to use RealSystem tools such as RealProducer™. For specific information about using a particular tool, refer to the tool's user's guide or online help.

- Writing SMIL

  To unify multiple clips into a single presentation, you use Synchronized Multimedia Integration Language (SMIL), a mark-up language that you can write with any text editor. If you've written HTML, you'll find it easy to pick up SMIL. To get started, turn to "Part III: Learning SMIL" beginning on page 125.

- Delivering a Presentation

  Once you finish production, you'll want to show off your work! "Part VII: Streaming Your Clips" beginning on page 393 explains how to stream your presentation from RealServer or a Web server, as well as how to embed it in a Web page.

Because this guide concerns streaming media production and the RealSystem core clip types, it does not cover the following topics:

- running RealServer
- broadcasting on the Internet
- using the RealONE Player interface
- using RealONE Player Javascript and ActiveX methods
- digital rights management
- digitizing and editing video and audio files
- preparing digital music clips for download (rather than streaming) to RealONE Player
- producing audio and video clips in formats other than RealAudio® and RealVideo®

> **Tip:** Although this guide does not explain how to produce audio and video in formats such as MPEG, many of the tips given in Chapter 3 and Chapter 4 apply to audio and video production in general, regardless of the streaming format.

How to Download This Guide to Your Computer

RealNetworks makes this guide available in the following formats for download to your computer:

- The HTML+Javascript version is available as a single, zipped archive that includes many samples that you can play in RealONE Player. It is highly recommended for persons who want to learn SMIL markup. You can read this version with Netscape Navigator or Microsoft Internet Explorer.
- The HTML Help version is available as a single `.chm` file for Windows 98 and later operating systems. It is identical to the HTML+Javascript version, except that it does not contain any sample files. The HTML Help version is smaller in size than the HTML+Javascript version, and it includes a search function.
- An Adobe Acrobat (PDF) version includes page numbers in cross-references, making it more useful than the HTML versions when printed. You can download the free Acrobat viewer from Adobe's Web site at **http://www.adobe.com/products/acrobat/readstep.html**.

All of the online versions of this guide are available for individual download from RealNetworks' Technical Support Web site at:

http://service.real.com/help/library/encoders.html

How this Guide Is Organized

Part I: Getting Started with Streaming Media

Whether you are new to streaming media, or an old hand, be sure to read the following chapters.

Chapter 1: New Features

If you're familiar with previous versions of RealSystem, this chapter will give you a quick update on the many changes in this version of RealSystem.

Chapter 2: Presentation Planning

If you are new to streaming media, this chapter walks you through the steps involved in putting together a RealSystem presentation, explaining bandwidth and timeline issues.

Part II: Producing RealSystem Clips

The clip is the basic unit of a streaming media presentation. The following chapters explain production issues for the core RealSystem clip types.

Chapter 3: Audio Production

This chapter gives you the background you need to create a RealAudio file. It then provides pointers on capturing and digitizing high-quality audio files.

Chapter 4: Video Production

Read this chapter to learn how to capture high-quality video content and optimize it for conversion to streaming RealVideo clips.

Chapter 5: Flash Animation

Using Macromedia's Flash, you can produce dazzling animated presentations. This chapter explains how to stream Flash clips with RealSystem.

Chapter 6: RealText Markup

With RealText®, you can create text that displays at different times in your presentation. This is a great way to provide video credits and subtitles, for example.

Part III: Learning SMIL

SMIL is the heart of streaming media, letting you pull together simple or highly complex presentations. Read the following chapters to get started.

Chapter 7: SMIL Basics

After you create your multimedia clips, you write a SMIL file that pulls the entire presentation together. This chapter explains the basic structure and syntax of a SMIL file.

Chapter 8: Clip Source Tags

This chapter explains how to add clips to a SMIL presentation, explaining the various streaming and download protocols, such as RTSP and HTTP.

Part IV: Organizing a Presentation

When you stream multiple clips, you use SMIL to group your clips and lay out the presentation. The following chapters explain how to organize your media.

Chapter 9: Presentation Information

This chapter demonstrates how to add presentation information to a SMIL file to enhance the playback experience and aid viewer accessibility.

Chapter 10: Groups

This chapter shows you how to make clips play together or in a sequence. Creating groups is the most basic way to set up a SMIL timeline.

Chapter 11: Layout

When clips play in parallel, you create a layout as described in this chapter. You can even make new clips pop up in new windows.

Part V: Timing and Linking Clips

Unlike a static Web page, streaming media *flows*. Timing is a key aspect of SMIL, and the following chapters explain how to create a timeline, as well as how to link to other resources.

Chapter 12: Basic Timing

The SMIL timing commands give you a powerful means to coordinate clip playback. Read this chapter to learn the basics of how to use SMIL to modify a presentation's timeline.

Chapter 13: Advanced Timing

Once you've mastered the basics of SMIL timing as described in Chapter 12, you're ready to learn about the advanced SMIL timing features described here.

Chapter 14: Hyperlinks

Refer to this chapter to learn how SMIL's hyperlinking capabilities let you launch new clips or presentations.

Part VI: Mastering Advanced SMIL Features

SMIL is a powerful language that lets you add special effects to your presentation. You can also use SMIL to manage bandwidth, and stream different clips to different viewers.

Chapter 15: Transition Effects

SMIL provides over a hundred special effects you can use when a clip starts or stops playback. This chapter shows you how to create eye-catching transitions.

Chapter 16: Animations

Read this chapter to learn how to use SMIL animations (not to be confused with Flash animation) to create special effects while clips play.

Chapter 17: Switching

SMIL lets you stream different presentations based on viewer criteria, such as available bandwidth or language preference. Read this chapter to learn about SMIL's switching capabilities.

Chapter 18: Prefetching

Prefetching is a powerful feature that lets you download clip data before a clip plays. This can help prevent presentation rebuffering.

Part VII: Streaming Your Clips

When you finish production, you're ready to make your presentation available for viewing as described in the following chapters.

Chapter 19: Web Page Embedding

If you want to integrate your presentation seamlessly into your Web page, follow the instructions in this chapter.

Chapter 20: Presentation Delivery

This chapter gives instructions for moving your streaming clips to RealServer and linking your Web page to them. It also explains how to use a Web server to deliver simple presentations.

Part VIII: Appendixes

The following appendixes gather useful information that will help you whether you're a novice or a professional.

Appendix A: Basic Questions

If you are new to RealSystem, this appendix answers basic production questions and points you to additional resources on the Internet.

Appendix B: Production Tasks

Consult this appendix when you want to carry out a specific production task, but don't know where to find the answer in this guide.

Appendix C: SMIL Tag Reference

Once you understand SMIL, use this appendix as a reference for SMIL tag and attribute values.

Appendix D: SMIL Color Values

This appendix covers the types of values you can use with SMIL color attributes.

Appendix E: RealText Tag Reference

This appendix provides a quick reference to RealText markup, which is explained in Chapter 6.

Appendix F: File Type Reference

This appendix provides a quick reference for common file types used in RealSystem streaming.

Appendix G: Language Codes

If you create streaming clips in different languages as described in Chapter 17, you use these codes in your SMIL file to indicate the language choices.

Conventions Used in this Guide

The following table explains the typographical conventions used in this production guide.

Notational Conventions

Convention	Meaning
emphasis	Bold text is used for in-line headings, user-interface elements, URLs, and e-mail addresses.
terminology	Italic text is used for technical terms being introduced, and to lend emphasis to generic English words or phrases.
`syntax`	This font is used for fragments or complete lines of programming syntax (markup).
`syntax emphasis`	Bold syntax character formatting is used for program names and to emphasize specific syntax elements.
`variables`	Italic syntax character formatting denotes variables within fragments or complete lines of syntax.
`[options]`	Square brackets indicate values you may or may not need to use. As a rule, when you use these optional values, you do not include the brackets themselves.
`choice 1|choice 2`	Vertical lines, or "pipes," separate values you can choose between.
…	Ellipses indicate nonessential information omitted from examples.

Additional RealSystem Resources

Most RealNetworks® manuals are available in both PDF and HTML formats from the RealNetworks documentation library. The library's main page is at **http://service.real.com/help/library/index.html**. In addition to this production guide, you may need the following resources:

- *RealPix Authoring Guide*

  With RealPix™ you can create streaming slideshows of still images. *RealPix Authoring Guide* explains how to put a RealPix presentation together with special effects such as fades and zooms. You can find this guide at **http://service.real.com/help/library/encoders.html**.

- *RealSystem Server Administration Guide*

  The basic reference for the RealServer administrator, this guide explains how to set up, configure, and run RealServer to stream multimedia. You need this guide only if you are running RealServer yourself. It is available at **http://service.real.com/help/library/servers.html**.

- *Embedded RealPlayer Functionality Guide*

  This guide supplements *RealSystem Production Guide*. Available at **http://service.real.com/help/library/encoders.html**, it explains how to use JavaScript or VBScript to control RealPlayer functions for a presentation embedded in a Web page.

- RealSystem Authoring Kit

  The authoring kit contains production tools and manuals in a single, convenient bundle. You can obtain the kit by registering at **http://www.realnetworks.com/products/authkit/index.html**.

- Software Development Kits (SDKs)

  RealNetworks offers SDKs for RealServer, RealPlayer, and RealProducer. Designed for programmers, SDKs help you integrate applications with RealSystem, or create new plug-ins for RealServer and RealPlayer. You can get SDKs by registering at **http://www.realnetworks.com/devzone/downlds/index.html**.

Technical Support

To reach RealNetworks' Technical Support, please fill out the form at:

- **http://customerrelations.real.com/scripts/rnforms/contact_tech_service.asp**

The information you provide in this form will help Technical Support personnel respond promptly. For general information about RealNetworks' Technical Support, visit this Web page:

- **http://service.real.com/help/call.html**

PART

I

GETTING STARTED WITH STREAMING MEDIA

Whether you are new to streaming media, or an old hand, this section will get you started with the latest RealSystem technology. Chapter 1 describes the many changes in this version of RealSystem, while Chapter 2 explains the basics of putting together a streaming presentation.

CHAPTER

1

NEW FEATURES

The open, end-to-end architecture of RealSystem iQ gives you more possibilities for creating Web-based multimedia than ever. If you're familiar with previous versions of RealSystem, this chapter gives you a quick look at changes in the latest release of RealSystem iQ.

RealONE Player Introduced

The successor to RealPlayer 8, RealONE Player provides the most advanced media playback possibilities available, combining streaming media playback with digital downloading. RealONE Player is available for Windows, Macintosh, and several Unix operating systems, including Linux.

SMIL 2.0 Support

RealONE Player supports SMIL 2.0, which adds many new features to SMIL 1.0. RealONE Player is backwards-compatible with SMIL 1.0, so it can play any existing RealSystem SMIL presentation. RealPlayer G2, RealPlayer 7, and RealPlayer 8 cannot play SMIL 2.0 presentations. These versions of RealPlayer autoupdate to RealONE Player before playing a SMIL 2.0 file. See Chapter 7 for basic SMIL 2.0 information.

> **Note:** This manual describes SMIL 2.0 only. For SMIL 1.0 information, see *RealSystem iQ Production Guide* for Release 8. This manual is available in HTML and PDF formats at **http://service.real.com/help/library/encoders.html**.

SMIL 2.0 Files Require an XML Namespace

A simple `<smil>` tag designates a SMIL 1.0 file. To write a SMIL 2.0 file, you need to include an XML namespace like this:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
```

For More Information: See "The SMIL 2.0 Tag and Namespace" on page 134 for more information.

SMIL 2.0 Attributes Use "Camel Case"

In SMIL 2.0, most attributes and predefined values that have multiple words now use camel case, in which all words are compounded and words following to the first word are capitalized. For example, the `system-bitrate` attribute in SMIL 1.0 becomes `systemBitrate` in SMIL 2.0. For more information, see "Tags, Attributes, and Values" on page 135.

New and Updated SMIL Resources

In addition to the chapters that describe SMIL 2.0 features, this manual adds or updates several resources that will help you with creating presentations:

- For information on updating SMIL 1.0 syntax to SMIL 2.0 standards, see "Updating SMIL 1.0 Files to SMIL 2.0" on page 143.
- Appendix B addresses specific production questions by referring you to the appropriate section in this manual.
- Appendix C provides a quick reference for SMIL tags and attributes that will help you once you're familiar with SMIL markup.
- Appendix D explains the types of color values that you can specify.
- Get the bundled HTML version of this production guide as described in "How to Download This Guide to Your Computer" on page 5. Then choose **Sample Files** from the **Go To** menu to display links to SMIL files that you can play in RealONE Player.

New Clip Tag Attributes

Chapter 8 explains the changes to SMIL clip source tags, such as `<video/>` tags. SMIL 2.0 introduces several new clip attributes, as the following sections explain.

Color Attributes

As described in "Modifying Clip Colors" on page 157, RealONE Player supports new color attributes that allow you to make the following transparent or semi-transparent:

- any color or range of colors in a clip

- all colors in a clip
- a clip's background color

Image Streaming Rates

SMIL 2.0 provides a new method for setting streaming rates for static clips such as images. See "Setting a Clip's Streaming Speed" on page 146.

Descriptive Metatags

In addition to title, author, copyright, and abstract attributes, clip source tags can have `alt`, `longdesc`, and `readIndex` attributes. These attributes allow assistive devices to read clip information for visually impaired viewers. For more information, see "Adding Accessibility Information" on page 169.

Colored Objects with the <brush/> Tag

The `<brush/>` tag functions just like a clip source tag such as `<video/>`. It does not link to a media clip, however. Instead, it defines a color that displays in a region. For more information, see "Creating a Brush Object" on page 148.

Expanded Grouping Possibilities

Chapter 10 explains `<seq>` and `<par>` groups, which have changed little from SMIL 1.0. It also covers the `<excl>` tag, which is new in SMIL 2.0.

Sequences Act Like a Single Presentation

In SMIL 2.0, a simple sequence of clips defined in a `<seq>` group acts like a single presentation instead of a series of separate presentations. See "Playing Clips in Sequence" on page 175 for more information.

New <excl> Groups

The exclusive group is a powerful feature that you can use to add interactivity to a presentation. The new `<excl>` tag creates an exclusive group, in which only one clip can play at a time. Unlike with a `<seq>` group, though, you can specify the order in which the `<excl>` group members play, have them interrupt each other, and select them based on any criteria, including mouse clicks. See "Creating an Exclusive Group" on page 186 for more information.

Synchronizing Parallel Elements

The section "Synchronizing Playback in Parallel Groups" on page 178 explains how to control which clips in parallel groups stay synchronized if bandwidth drops. This advanced feature also lets you create an independent timeline for a clip to make it act like a broadcast. In this case, viewers cannot rewind or fast-forward through the clip.

Enhanced Layout Choices

Chapter 11 explains how to lay out clips in RealONE Player using SMIL 2.0 layout tags and attributes, which provide many new layout possibilities. The following sections describe the principal new features of SMIL 2.0 layout.

Secondary Pop-Up Windows

You can now use `<topLayout>` tags to create secondary media windows that pop up during a presentation. This window is useful for playing supplemental clips, or clips that do not fit the main media window's layout. See "Secondary Media Windows" on page 197 for more information.

Subregions

The section "Subregions" on page 196 describes how to create regions within regions. Creating a subregion is useful if you want to associate a smaller region with a larger region so that the smaller region changes position automatically if you reposition the larger region.

Region Size and Position Attributes

To set a region's size and position within a window, you can now use `bottom` and `right` attributes, as well as `height`, `width`, `left`, and `top`. You can also use any combination of these attributes to create a region, giving you more ways to define layouts. See "Defining Region Sizes and Positions" on page 208 for more information.

Registration Points

With registration points, which are described in the "Clip Position and Fit" on page 198, you can easily position clips within large regions. You can use a registration point to center clips, for example, or align them to a region's lower-right corner.

Region Transparency

RealONE Player supports true region transparency, meaning that clips behind transparent areas of another clip are visible. See "Transparency in Regions and Clips" on page 217 for more information.

Backgrounds Can Appear Only When Clips Play

You can now set region background colors to appear only when a clip plays in the region. Previously, all regions and backgrounds appeared automatically at the start of the presentation. See "Setting When Background Colors Appear" on page 216 for more information.

More Timing Possibilities

SMIL 2.0 provides many new ways to construct presentation timelines. Chapter 12 and Chapter 13 explain basic and advanced timing attributes, respectively.

New Element Repeat Attributes

The new `repeatCount` and `repeatDur` attributes replace the SMIL 1.0 `repeat` attribute. The new attributes let you specify a total number of repetitions, or the total length of the repeating cycle, respectively. See "Repeating an Element" on page 247 for details.

Additional fill Values

The `fill` attribute includes new values (`auto`, `default`, `hold`, and `transition`) that let you specify additional `fill` behaviors. A new `fillDefault` attribute lets you set the `fill` behavior for entire groups. For more information, see "Setting a Fill" on page 251, as well as "Specifying a Default Fill" on page 258.

Advanced Timing Values

See Chapter 13 for information on advanced SMIL timing values. These values work with the `begin` and `end` attributes to start and stop elements when certain events occur, such as when the viewer clicks a clip or presses a keyboard key.

New Linking Attributes

Chapter 14 covers hyperlinking, explaining how to link a SMIL presentation to a Web page or another SMIL presentation.

The <area/> Tag Replaces <anchor/>

The SMIL 2.0 <area/> tag replaces the SMIL 1.0 <anchor/> tag. The <area/> tag lets you turn an entire clip into a link, as well as create hot spots (image maps) over a clip. You can now create hot spots as rectangles, circles, and polygons. See "Using the <area/> Tag" on page 283.

New Ways to Open Links

SMIL 2.0 includes new ways to specify when a link opens. You can define a keyboard key that the viewer can press to open a link, for instance, or you can make links open automatically at any point in the presentation. See "Defining Basic Hyperlink Properties" on page 290 for more information.

Ability to Control RealONE Player State

The SMIL attribute sourcePlaystate in a link controls RealONE Player's state when a link is clicked. You can make the RealONE Player presentation pause, stop, or continue playing when the link opens. For more on sourcePlaystate, see "Linking to HTML Pages" on page 294 or "Linking to Streaming Media" on page 301.

RealONE Player Browsing Windows

RealONE Player on Windows includes its own browsing windows, which allow you to display HTML pages within the RealONE Player environment, as well as in the viewer's default Web browser. The RealONE Player context window, which appears to the right of the media window, can display HTML pages that supplement the streaming presentation. For more information, see "Linking to HTML Pages" on page 294.

Clip Transition Effects

Chapter 15 explains how to define clip transition effects, which are special effects that display when a clip starts or stops. You can use transition effects to crossfade sequential clips, for example, or introduce a new clip with a slide, a wipe, or over a hundred other effects found in professional video production.

SMIL Animations

Chapter 16 explains how to create SMIL animations, which are special effects that occur as a clip plays. You can use SMIL animations to shrink a clip, move

it around the screen, alter its background color, and change its volume level, for example. Unlike Flash animation, SMIL animations are not clips. Rather, they are instructions that tell RealONE Player how to alter the display of other clips, whether videos, still images, audio clips, or so on.

Powerful Content Control Capabilities

SMIL 2.0 has sophisticated content control features that allow the advanced SMIL author to tailor presentations for different audiences and network conditions.

Additional Switching Test Attributes

SMIL 2.0 includes several new test attributes that you can use in <switch> groups. These attributes let you display alternative presentations for different monitor sizes or operating systems, for example. For a list of test attributes, see "Available Test Attributes" on page 366.

Inline Switching

You can add any switching test attribute directly to a clip source tag or a group tag without using a <switch> tag. RealONE Player then plays the clip or group only if it satisfies the attribute value. Although not recommended for all situations in which switching is required, inline switching can be useful in many cases. For more information, see "Using Inline Switching" on page 365.

Prefetching Clip Data

With <prefetch/> tags, you can download clip data before clips play. This feature gives you a powerful way to manage your presentation's streaming bandwidth. See Chapter 18 for information on prefetching.

Additions and Deletions to this Guide

In addition to describing new features of RealSystem, this version of the *RealSystem iQ Production Guide* includes several organizational changes from previous versions.

RealText Markup Described

Chapter 6 of this guide includes the latest information about RealText, including how to turn a RealText clip's background transparent or semi-

transparent. For more on RealText transparency, see "Creating a Transparent Window Background" on page 96.

> **Note:** The *RealText Authoring Guide* is still available, but it covers RealText only through RealSystem iQ Release 8.

No Chapters on Advertising and Broadcasting

This guide no longer contains chapters that cover the RealSystem Advertising Application and broadcasting. Information about using SMIL with the Advertising Application is available separately. For information about broadcasting media, see *RealProducer User's Guide* and *RealServer Administration Guide*.

CHAPTER

2

PRESENTATION PLANNING

A streaming presentation can consist of one or many clips. No matter how simple or complicated your presentation, you'll need to plan your media production so you can work effectively and reach your target audience. This chapter explains the basics of how to put streaming media presentations together. If you are not yet familiar with RealSystem components, see also Appendix A beginning on page 435.

Step 1: Decide How to Deliver Clips

The first step in creating a streaming presentation is to consider the last step: how will you deliver your clips to other people? How you plan to stream your clips can greatly affect your media production.

RealServer Streaming

RealServer is the preferred host for RealSystem presentations. Designed specifically to stream multimedia over networks, RealServer keeps multiple clips synchronized and uses many advanced features to ensure that clips stream smoothly, even under adverse network conditions. A RealServer administrator sets up and runs each RealServer. If you will not be running RealServer yourself, check the following with your RealServer administrator:

1. What version of RealServer is available?

   To deliver clips described in this manual, you'll need RealServer G2, 7, or 8, rather than any of the earlier versions. Make sure that your RealServer can deliver all the clips you plan to develop.

2. How many streams can RealServer serve?

   Each RealServer has a maximum number of media streams it can send out at once. A RealServer with a maximum of 500 streams, for example, can

stream video to 500 viewers simultaneously. Make sure that the RealServer you plan to use has adequate capacity for your needs.

3. Are there any bandwidth constraints?

   The RealServer computer may lack the outgoing bandwidth to deliver a lot of high-speed clips simultaneously. If you plan to develop high-bandwidth presentations, confer with the RealServer administrator about bandwidth limitations.

4. Where will your clips reside?

   Your clips typically reside on RealServer, whereas your Web pages are on a Web server. You'll need to know the URLs for your clips on RealServer so that you can set up your Web page hyperlinks correctly.

5. Do any RealServer features need to be set up?

   The RealServer administrator can set up many streaming and security features, such as:

   - Live broadcasts
   - Pay-per-view content
   - Automatic ad insertion
   - Password authentication

Using RealServer through an Internet Service Provider

If an Internet service provider (ISP) hosts your Web pages, contact the ISP administrator to check out the RealServer issues described above. Also find out how much disk space you will have for streaming media. Many ISPs allot you a certain amount of disk space on their servers, such as 5 or 10 MB. Although this is a generous amount for Web pages, it's not much for streaming media. A single video clip can easily take up that much space.

Web Server Downloading

Although Web servers can deliver some streaming clips, they don't have RealServer's ability to synchronize clips and keep long presentations flowing smoothly. When only a Web server is available, you can still deliver multimedia presentations, but you will not be able to use all of the features that RealSystem offers.

> **For More Information:** If you plan to deliver clips with a Web server, read "Limitations on Web Server Playback" on page 428.

Local Playback

You can also create presentations that play back from a user's local computer. An example of this is a multimedia-enhanced book written with HTML and containing links to RealSystem clips. Users download the files to their computers, playing back the media clips with RealONE Player. In this case, you produce clips as described in this production guide, except that you don't target specific network connection bandwidths. In the HTML pages, URLs point to clips on the user's computer instead of on RealServer.

> **For More Information:** For more on local URLs in SMIL files, see "Linking to Local Clips" on page 151. See also "Launching RealONE Player with a Ram File" on page 420.

Step 2: Choose Clip Types and Gather Tools

RealSystem gives you many possibilities for creating streaming media. Your presentation may consist of a single clip, or several clips that play together. As you decide what types of clips you want to stream, gather the production tools you'll need to make the clips.

> **Tip:** Download the HTML version of this guide as described in "How to Download This Guide to Your Computer" on page 5. That version contains clip samples you can view with RealONE Player. Visit **http://www.realstore.com** to find out about the many tools available to help you create streaming media.

Audio and Video

RealAudio and RealVideo are the most popular streaming media formats. To produce them, you run an encoding tool with audio or video input from a live source, a recording format such as tape or CD, or a digitized file in a standard format such as WAV, AVI, QuickTime, or MPEG.

Audio and Video Production Tools

A streaming RealAudio or RealVideo clip results from gathering, editing, and encoding audio or video input. To carry out the initial steps of gathering and editing content, you'll need the following:

- A video camera and a microphone

  To capture live input, use any video camera and microphone that can be attached to your computer. You will not need these devices, though, if your audio or video source is already digitized.

- An audio/video capture card

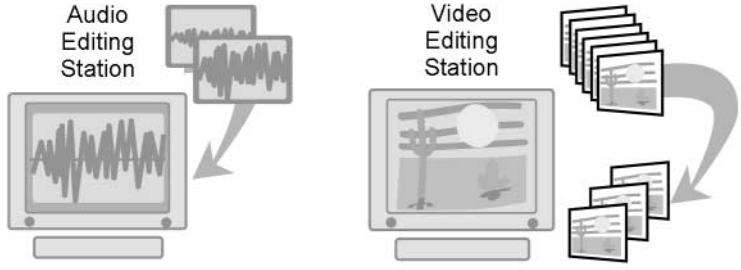
  To take input from a microphone or camera, your computer needs an audio/video capture card. This card accepts the input and digitizes it into a format you can edit. On Windows computers, you can use any video capture card that supports Video for Windows.

- Audio and video editing software

  These programs let you edit digitized audio and video files. When creating clips that stream on demand, it's best to edit and optimize the input before encoding it. When broadcasting, you can convert audio and video input into RealAudio or RealVideo clips directly from a capture card without first creating a separate, digitized file.

Producing RealAudio and RealVideo does not require that you use specific microphones, cameras, capture cards, or editing tools. Just ensure that your editing tools can save files in formats you can easily convert to streaming formats with your encoding tool.

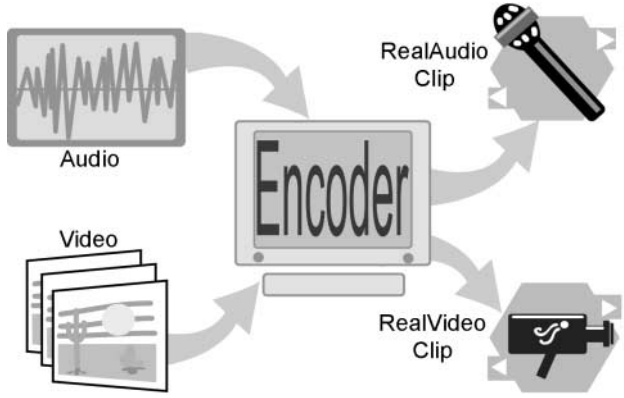
Use Your Favorite Editing Tools to Prepare Audio and Video Files



RealAudio and RealVideo Encoding Tools

Some editing programs can export digitized audio and video directly to RealAudio or RealVideo. If your editing program cannot export clips or you don't want to use this feature, you can use a RealNetworks tool to encode clips from files in standard formats such as WAV, AVI, QuickTime, and MPEG. RealProducer Basic is a free tool for encoding RealAudio and RealVideo clips. RealProducer Plus is an enhanced version that offers more encoding features.

RealProducer Creates Streaming Clips



For More Information: For more information about RealProducer, see "Getting Production Tools" on page 436. You can find tools at **http://www.realnetworks.com/products/index.html**.

SMIL

When you want to combine two or more clips into a single presentation, you use SMIL. Pronounced "smile," SMIL is a simple markup language that tells RealONE Player how to lay out and play your clips. You can use any word processor or text editor to write SMIL. For basic information about SMIL, see "Writing SMIL Files" on page 438. For details, see Chapter 7.

Animation

With Macromedia Flash, you can build anything from streaming cartoons to e-commerce applications. To create a soundtrack, you can encode a RealAudio clip that streams along with the Flash clip. See Chapter 5 for details about

producing Flash animation for RealONE Player. Learn more about Flash from Macromedia's Web site at:

http://www.macromedia.com/software/flash/

Images

Streaming multimedia presentations played with RealONE Player can include still images in the following formats:

- GIF87, GIF89, and animated GIF (`.gif`)

  Both interlaced and noninterlaced GIFs will work, but noninterlaced GIFs are recommended.

- JPEG (`.jpg`)

  RealONE Player can display RGB baseline JPEGs. Progressive and grayscale JPEGs are not supported.

- PNG (`.png`)

  RealONE Player does not adhere to gamma settings in PNG images.

> **Note:** Image transparency is not supported between SMIL regions. For more on this, see "Making a Region Partially Transparent" on page 216.

Images in SMIL Presentations

To add images to streaming presentations as backgrounds or buttons, for example, simply incorporate the images by using SMIL. This way, you can specify exactly where images appear in relation to your clips. You can also use SMIL to turn images into hyperlinks.

RealSlideshow Presentations

When you want to create a streaming slideshow, the easiest solution is to use RealSlideshow™ or RealSlideshow Plus. These tools have drag-and-drop interfaces that let you quickly build your slideshow, which can include text captions, audio narrations, and background music. RealNetworks' partners will even host your streaming slideshow. Get RealSlideshow at:

http://www.realnetworks.com/products/index.html

RealPix Markup

Streaming slideshows are based on the RealPix markup language. Instead of using RealSlideshow, you can write your own markup to assemble images into a RealPix presentation that has eye-catching special effects such as dissolves and zooms. Learn the markup from *RealPix Authoring Guide*:

http://service.real.com/help/library/encoders.html

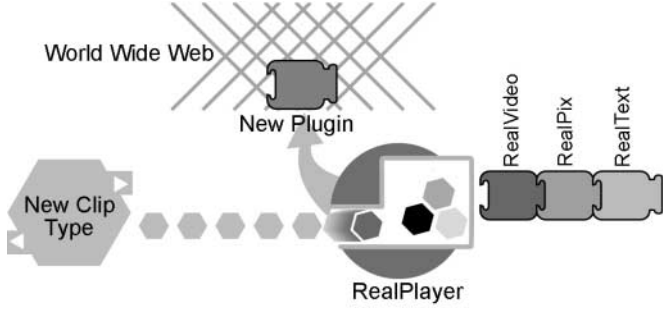
Text

Unlike HTML, SMIL does not display text directly. To show text in RealONE Player, you can add text to any image, video, or animation clip. Or you can use RealText, which streams text at specific times within a presentation. RealText lets you subtitle videos, for example, or create hypertext links. Chapter 6 explains the RealText markup.

Autoupdate Feature

RealONE Player's plug-in and autoupdate technologies ensure that your clips can reach the widest audience possible. RealONE Player plug-ins function like Web browser plug-ins. If RealONE Player doesn't have a plug-in needed to play a particular streaming clip, it downloads that plug-in from the Internet. RealONE Player can even use its autoupdate technology to upgrade itself to a new version when necessary.

RealONE Player **Downloads Plug-ins it Needs from the Internet**



Tip: RealONE Player's plug-in technology lets it play many types of clips developed by RealNetworks' partners. Check **http://www.realnetworks.com/devzone** for news about other types of streaming media.

Compatibility with Earlier Versions of RealPlayer

Plug-in and autoupdate technologies were introduced with RealPlayer G2. Earlier versions of RealPlayer cannot upgrade themselves, so they cannot play all the clips described in this production guide. Generally, you don't need to be concerned with backward compatibility because most RealPlayer users upgrade to the latest release. The following table summarizes which versions of RealPlayer offer which features. RealPlayer 4.0, for example, plays only RealAudio and RealVideo.

Supported Features in RealONE Player and Earlier RealPlayer Versions

Feature	RealONE	8	7	G2	5	4	3	2	1
RealAudio streaming	X	X	X	X	X	X	X	X	X
RealVideo streaming	X	X	X	X	X	X	–	–	–
Flash 2.0 streaming	X	X	X	X	X	–	–	–	–
Flash 3.0 and 4.0 streaming	X	X	–	–	–	–	–	–	–
RealPix streaming	X	X	X	X	–	–	–	–	–
RealText streaming	X	X	X	X	–	–	–	–	–
SMIL 1.0 presentations	X	X	X	X	–	–	–	–	–
SMIL 2.0 presentations	X	–	–	–	–	–	–	–	–
Plug-ins for additional clip types	X	X	X	X	–	–	–	–	–
Autoupdate	X	X	X	X	–	–	–	–	–

This table covers only general clip compatibility, not codec compatibility. The latest version of RealProducer creates RealAudio and RealVideo clips with codecs that RealPlayer 4 and earlier versions cannot play back. RealProducer has a compatibility option for RealPlayer 5, making that the earliest version of RealPlayer you can reach with current production methods.

Protection of Copyrighted Content

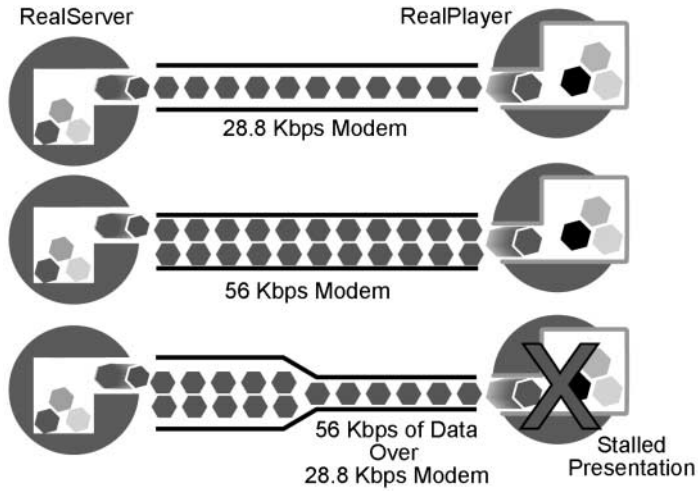
Unlike a Web browser, RealONE Player does not store clips in a disk cache or allow users to copy or download still images. This helps you keep copyrighted material secure when you stream clips from RealServer, though not from a Web server. A RealProducer option lets you encode RealAudio and RealVideo clips so that viewers can record the clips on their computers.

Step 3: Develop a Bandwidth Strategy

Any computer connected to a network has a connection bandwidth, which is a maximum speed at which it can receive data. Web users with 28.8 Kbps modems, for example, can view only those presentations that stream less than 28.8 Kb of data per second. Presentations that stream more data than that per second may stall because the data cannot get over the modems fast enough to keep the clips flowing. These presentations will not cause problems for users with faster connections, though.

Successfully targeting your audience's connection bandwidth is crucial for developing streaming media. Viewers don't like to wait more than a few seconds for playback to begin after they click a link. And if your clips sputter because they use too much bandwidth, viewers are not likely to stay tuned. Developing a bandwidth strategy helps ensure that clips play back quickly and don't stall. You can also devise ways to deliver good clips to users with slow connections, and great clips to those with fast connections.

Presentation Data Must Fit RealONE Player's Bandwidth



Buffering

For each streaming clip, RealONE Player keeps a "buffer" that acts as a data reservoir. Data enters the buffer as it streams to RealONE Player, leaving the buffer as RealONE Player plays the clip. The buffer helps ensure that lapses of available bandwidth don't stall the presentation. If network congestion halts the flow of data for a few seconds, for example, RealONE Player keeps the clip

playing with the buffered data. Your goal is to minimize initial buffering and eliminate rebuffering.

Initial Buffering (Preroll)

RealONE Player buffers a few seconds of data before a clip plays. Also called "preroll," initial buffering is required for every clip. Developing clips that use an appropriate amount of bandwidth keeps preroll to an acceptable level. You want preroll to be low—less than 15 seconds for each clip. RealAudio and RealVideo encoding tools set a low preroll for you. With other clips, though, how you create the clip determines its preroll.

Rebuffering

When clip data has stopped coming in and the clip buffer is empty, RealONE Player has to halt clip playback to store data again, or "rebuffer." Sometimes this is unavoidable because the viewer's available bandwidth drops for too long. When developing a multiclip presentation, though, you need to consider timelines carefully so that you don't inadvertently cause rebuffering, which can happen if too many clips fight for too little bandwidth.

Audience Bandwidth Targets

Your streaming presentations should never consume all of your audience's connection bandwidth. They must always leave bandwidth for network overhead, error correction, resending lost data, and so on. Otherwise, they may require frequent rebuffering. The following table recommends maximum streaming speeds for common network connections. To reach 28.8 Kbps modems, for example, a presentation should stream no more than 20 Kb of data per second.

Maximum Streaming Rates

Target Audience	Maximum Streaming Rate
14.4 Kbps modem	10 Kbps
28.8 Kbps modem	20 Kbps
56 Kbps modem	34 Kbps
64 Kbps ISDN	45 Kbps
112 Kbps dual ISDN	80 Kbps
Corporate LAN	150 Kbps
256 Kbps DSL/cable modem	225 Kbps

(Table Page 1 of 2)

Maximum Streaming Rates (continued)

Target Audience	Maximum Streaming Rate
384 Kbps DSL/cable modem	350 Kbps
512 Kbps DSL/cable modem	450 Kbps

(Table Page 2 of 2)

For any other connection speed, calculate the maximum streaming speed as:

- Approximately 75 percent of the connection bandwidth for analog connections such as dial-up modems.

  –Or–

- Approximately 90 percent of the connection bandwidth for high-speed digital connections such as DSL or cable modems.

Multiclip Presentations

When several clips are played together, their streaming speeds added together should not exceed the connection maximum. For example, RealPix and RealAudio clips streaming at 12 and 8 Kbps, respectively, can play in parallel over 28.8 Kbps modems because together they stream at 20 Kbps. However, they cannot play back together if they stream at 12 and 16 Kbps, respectively, because the 28 Kbps total streaming speed leaves the modem no bandwidth for overhead. Such a presentation would likely require frequent rebuffering.

Streaming at Less than the Maximum Speed

Your presentations do not have to stream at the maximum speeds listed in the preceding table. In some cases, you may want your clips to stream at less than the maximum:

- You may need to leave enough bandwidth for the user to perform other network activities. When streaming an Internet radio station, for example, leave some bandwidth for the listener to view Web pages.
- Bandwidth is shared by everyone on a local area network (LAN). If the LAN is heavily used, the 150 Kbps LAN target speed may slow down the LAN too much. For an intranet, the LAN manager should decide the maximum streaming rate.

Clip Bandwidth Characteristics

To reach your target audiences with your clips, you need to understand your clips' bandwidth characteristics.

RealAudio and RealVideo

A RealAudio and RealVideo encoding tool can turn your source audio or video file into a clip that streams to any target connection with little preroll. But if the tool has to squeeze a file down too much to reach a low-bandwidth target, clip quality may degrade. So although the clip will stream well, you might not like the results. To ensure good-quality playback, keep your streaming bandwidths in mind when creating source files, especially when you plan to reach dial-up modem users.

> **For More Information:** See "Understanding RealAudio" on page 43 and "Understanding RealVideo" on page 57.

Flash

Macromedia Flash streams well at low bandwidths, making it an attractive alternative to video. Low streaming speed doesn't affect Flash's visual quality as it can with video. At low bandwidths, though, you may not be able to include as many items in your animated scenes as when streaming at high bandwidths. After you develop a Flash clip for RealONE Player, you tune it to stream at a specific bit rate. For more on this, see "Flash Bandwidth Characteristics" on page 76.

RealText and SMIL

Because RealText and SMIL files are plain text, they use little bandwidth. You generally don't need to be concerned about how they affect a presentation's bandwidth consumption.

RealPix (Slideshows)

RealPix bandwidth use depends on the image sizes and how soon each image must appear in the clip's timeline. At higher bandwidths, you can use larger images and display them at shorter intervals. By varying image size and the RealPix timeline, you gain a lot of control over bandwidth use. RealSlideshow always ensures that images stream at a rate appropriate for your target audience. If you write RealPix markup by hand, though, you need to be careful not to overload a connection's bandwidth.

Images in SMIL Presentations

JPEG, GIF, or PNG images in a SMIL presentation stream at 12 Kbps. See "Setting a Clip's Streaming Speed" on page 146 for instructions on changing this streaming bit rate.

Reaching Multiple Audiences

To provide good content for users with slower connections, and great content for those with faster connections, you can use two methods, combining them if needed:

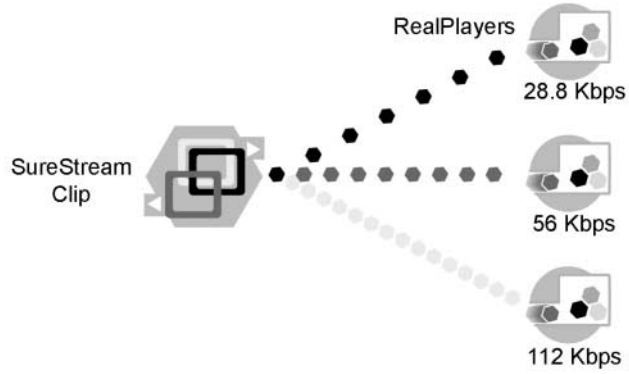
- Create a single RealAudio or RealVideo clip that targets different audience bandwidths by using SureStream technology.
- Create separate clips for each bandwidth target, and let RealONE Player choose which set of clips to play through SMIL.

Either way, you add to your Web page just one link for all visitors. You don't need separate links for modems and DSL connections, for example.

SureStream RealAudio and RealVideo

With RealSystem's SureStream technology, you can encode a RealAudio or RealVideo clip for multiple bandwidths. For example, you can encode a single RealAudio music clip for 28.8 Kbps modems, 56 Kbps modems, 112 Kbps dual ISDN, 256 Kbps DSL, and so on. The clip's playback quality improves with each faster speed. When a viewer clicks a link to a SureStream clip, RealONE Player and RealServer determine which stream to use based on the available bandwidth, as shown in the following illustration.

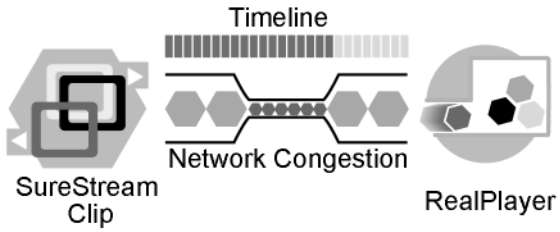
SureStream Clip Encoded for Multiple Bandwidths



RealServer and RealONE Player can even adjust this choice to compensate for network conditions. If a fast connection becomes bogged down because of high network traffic, RealServer switches to a lower-bandwidth stream to prevent the presentation from stalling. When the congestion clears, RealServer

switches back to the higher-bandwidth stream. RealONE Player doesn't need to rebuffer data during this shifting.

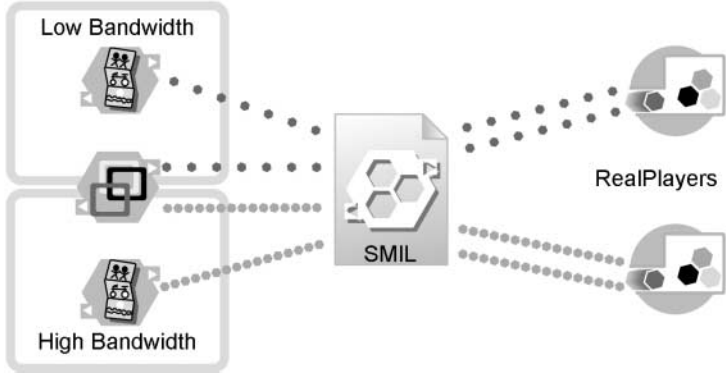
Switching Bandwidths During Network Congestion



Switching Between Multiple Clips with SMIL

Only RealAudio and RealVideo clips can stream at multiple bandwidths. You can create multiple versions of other clips, though, each for a different bandwidth. RealONE Player then chooses which clip to play based on a SMIL bandwidth parameter. The following illustration shows a SMIL file that lists separate high-bandwidth and low-bandwidth RealPix clips. Each RealONE Player evaluates the SMIL file and chooses the RealPix clip appropriate for its connection speed. Both presentations use the same SureStream RealAudio clip, though, which has been encoded internally for multiple bandwidths.

Bandwidth Choices through a SureStream Clip and SMIL



When you use SMIL for bandwidth choices, RealServer cannot downshift to a lower-bandwidth clip group the way it can downshift to a slower SureStream stream. RealServer employs other techniques, though, to compensate for network congestion. Its stream thinning capabilities enable it to drop low-priority data to decrease the presentation bandwidth temporarily. When the

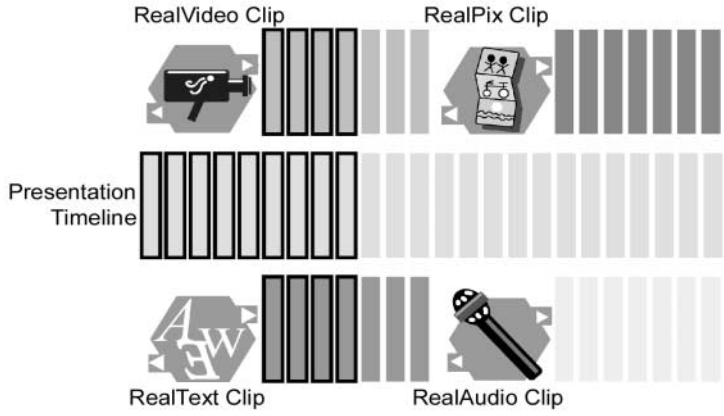
network congestion clears, RealServer continues to stream all the presentation data.

> **For More Information:** "Switching Between Bandwidth Choices" on page 370 explains how to use SMIL to designate different bandwidth groups.

Step 4: Organize the Presentation Timeline

Every streaming media clip has a timeline. A RealAudio clip may play for five minutes, for example, giving it a five-minute timeline. When clips are streamed together, you have a presentation timeline as well. Before producing clips, plan the presentation timeline. Among other things, the timeline can determine the order in which you produce clips. A well-conceived timeline also helps ensure that clips do not overload a connection's bandwidth and cause rebuffering.

Clip Timelines Coordinate with a Presentation Timeline



Timeline Considerations

When you assemble a streaming media presentation, you can manipulate various aspects of clip timelines.

Clips with Internal Timelines

Audio, video, and animation have internal timelines. In a 10-minute video, for instance, each frame corresponds to a specific point in a 10-minute timeline. Each second of audio meshes with each second of the image throughout the

clip's overall timeline. Your video, audio, or animation software is your main tool for manipulating the clip's timeline, which is woven into the fabric of the clip.

Clips with Variable Timelines

With RealPix or RealText, you use the markup language to control when each image or text block appears and how long it lasts. When combining clips, it's typically easier to produce audio, video, or animation first. Then set the RealPix and RealText timelines to coordinate with those clips.

SMIL Timing Commands

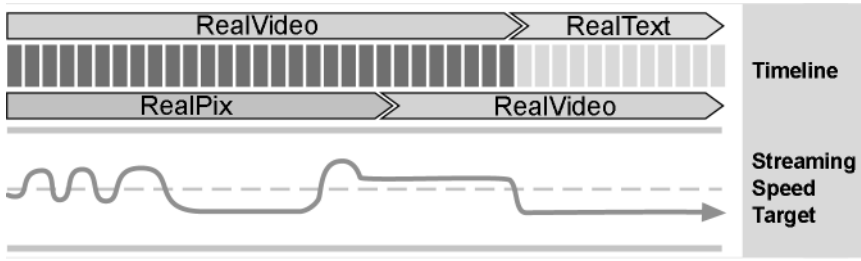
A SMIL file can include its own timing elements. Timing a presentation with SMIL can be as simple as having one clip start as soon as another one stops. But you can also use commands to delay playback for 10 seconds, for example, or to have a clip start playing 30 seconds into its internal timeline. SMIL's timing commands are optional, but they give you the flexibility you may need for some presentations.

For More Information: For more on SMIL timing, see "Specifying Time Values" on page 239.

Timelines for Multiclip Presentations

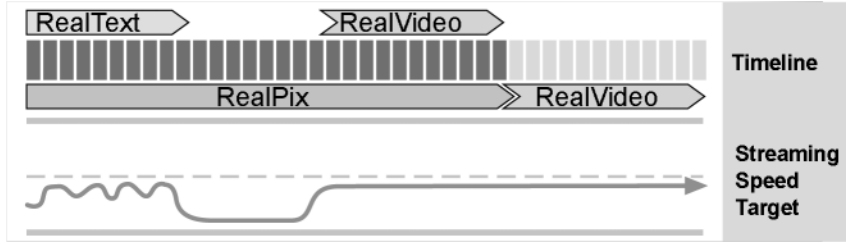
For presentations that include multiple clips, consider how to group clips without overloading an audience's connection bandwidth. The following illustration shows poor timeline planning. At various points, RealVideo and RealPix clips playing together exceed the connection's maximum streaming speed, which is represented below by the dashed line. Illustrated by the solid line, bandwidth use peaks again when the second RealVideo clip begins to play before the first video clip finishes. This presentation requires a high preroll for clips and would likely result in rebuffering at peak points.

Poor Bandwidth Use in a Multiclip Presentation



The next illustration shows better timeline planning and bandwidth management. The presentation starts with a low-bandwidth RealText clip that does not interfere with the streaming of the images in the RealPix clip. A RealVideo clip starts after the RealPix clip has streamed all of its images and does not need any more bandwidth. The second RealVideo clip starts after the first RealVideo clip has ended, so the two clips do not compete for bandwidth.

Improved Bandwidth Use in a Multiclip Presentation



Timeline Management

When developing a streaming presentation, keep the following in mind:

- Consider the presentation timeline carefully to eliminate bandwidth bottlenecks. These typically occur when two or more high-bandwidth clips play simultaneously. You may need to omit high-bandwidth pairings, combining high-bandwidth clips with low-bandwidth clips instead.
- Stagger the start times for clips. Every clip requires a certain amount of preroll before RealONE Player can play it. Your presentation will flow more smoothly if RealServer does not need to send more than one clip's preroll at a time.
- Start presentations with low-bandwidth clips. For example, use RealText to display credits. Or begin with a highly compressed RealAudio narration

before bringing in any other clips. RealSystem can take advantage of the extra bandwidth to begin streaming higher-bandwidth data to RealONE Player "behind the scenes."

- Test your presentations in "real world" circumstances, replicating your audience's bandwidth conditions. Clips may play back OK from your desktop computer but bog down when streamed over a modem.

Step 5: Create Your Clips

When you've decided how you'll stream clips, chosen clip types and tools, developed a bandwidth strategy, and planned a timeline, you're ready to start creating streaming presentations. The rest of this guide provides production pointers, but does not explain how to use any specific tools. Be sure to have the documentation for your production tools handy as you develop your clips.

- If you're producing audio, video, or animation, read Chapter 3, Chapter 4, or Chapter 5, respectively.
- For information on SMIL, start with Chapter 7.
- To add clips to a Web page, follow the instructions in Chapter 19.
- When you're ready to move clips to servers, read Chapter 20.

PART

II

PRODUCING REALSYSTEM CLIPS

Individual clips are the building blocks of a streaming media presentation. In this section, Chapter 3 and Chapter 4 take on RealAudio and RealVideo production, respectively. Read Chapter 5 to learn how to stream Macromedia's Flash animation. With RealText, which Chapter 6 covers, you can create timed text that displays alongside other clips.

CHAPTER

3

AUDIO PRODUCTION

RealNetworks pioneered streaming audio with RealAudio, the first streaming media product for the Internet. Since its debut in 1995, RealAudio has become the standard for network audio, delivering stereo sound over 28.8 Kbps modems and CD-quality sound at high connection speeds. This chapter gives pointers on how to prepare and encode your sound files for streaming.

Understanding RealAudio

Because RealAudio clips are compressed, you typically start with a sound file in a digitized, uncompressed format such as WAV or AIFF. Using a RealAudio encoding tool, you create a RealAudio clip from the source file. RealAudio clips use the file extension `.rm`, although older clips may use `.ra` instead. This section explains how RealAudio encodes an audio file for streaming. This knowledge will help you produce high-quality streaming clips.

Bandwidth and Audio Quality

One way that RealAudio squeezes an audio file's size down is by throwing out nonessential data. This makes it a *lossy* compression format. RealAudio doesn't delete data indiscriminately, though. It first jettisons portions you cannot hear, such as very high and very low frequencies. Next, it removes as much data as needed while keeping certain frequencies intact. Voice encoding favors frequencies in the normal human speaking range. Music encoding retains a broader frequency range.

Although RealAudio is savvy about what audio data it throws out, be aware that the lower the connection speed, the more data gets ejected, and the cruder the sound quality becomes. At low bandwidths, you get roughly the quality of an AM radio broadcast. With faster connections, you can encode music with FM-quality sound. And at the high speeds of DSL, cable modems, and LANs, RealAudio sound quality rivals that of CD playback. When creating

RealAudio clips for low bandwidths, it's important to start with high-quality input, as described in "Capturing Audio" on page 50, to attain good sound quality.

RealAudio Bandwidth Characteristics

You create a RealAudio clip by using one or more RealAudio *codecs*. A codec is a coder/decoder. It tells an encoding tool how to turn audio source files into RealAudio clips. On the receiving end, RealONE Player uses codecs to expand clips into audio data the computer can play. RealAudio employs a series of codecs, each of which creates an audio stream for a precise bandwidth. One codec compresses mono music for a 28.8 Kbps modem. Another one compresses stereo music for that same modem speed. This set of codecs is different from the set used to compress music for, say, DSL and cable modem connections.

A RealAudio clip consumes bandwidth at a flat rate determined by the codec used to encode the clip. A RealAudio clip encoded with a 20 Kbps codec, for example, steadily consumes 20 Kbps of bandwidth as it plays. The following table lists the standard bit rates for RealAudio clips encoded for specific target audiences by RealProducer 8.5. Encoding a voice-only audio file for a 28.8 Kbps modem, for example, creates a 16 Kbps streaming clip. With mono music input, though, you get a 20 Kbps clip.

RealAudio Standard Bit Rates

<table>
<tr><th>Target Audience</th><th>Voice Only</th><th>Voice and Music</th><th>Mono Music</th><th>Stereo Music</th></tr>
<tr><td>28.8 Kbps modem</td><td rowspan="2">16 Kbps</td><td>20 Kbps</td><td>20 Kbps</td><td>20 Kbps</td></tr>
<tr><td>56 Kbps modem</td><td>32 Kbps</td><td>32 Kbps</td><td>32 Kbps</td></tr>
<tr><td>64 Kbps single ISDN</td><td>32 Kbps</td><td>44 Kbps</td><td>44 Kbps</td><td>44 Kbps</td></tr>
<tr><td>112 Kbps dual ISDN</td><td rowspan="3">64 Kbps</td><td>64 Kbps</td><td rowspan="4">64 Kbps</td><td>64 Kbps</td></tr>
<tr><td>Corporate LAN</td><td rowspan="3">96 Kbps</td><td>132 Kbps</td></tr>
<tr><td>256 Kbps DSL/cable modem</td><td>176 Kbps</td></tr>
<tr><td>384 Kbps DSL/cable modem</td><td rowspan="2">96 Kbps</td><td>264 Kbps</td></tr>
<tr><td>512 Kbps DSL/cable modem</td><td>352 Kbps</td></tr>
</table>

In terms of bandwidth use, RealAudio is the most *in*flexible media type. The RealAudio codecs set streaming bit rates in a stairstep model: 20 Kbps, 36 Kbps, 44 Kbps, and so on, with no inbetween choices. Because RealAudio clips

always stream at specific bit rates, consider their bandwidth needs first when you use them in multiclip presentations. Then create your other clips to stream within the bandwidth that's left.

> **Note:** With SureStream technology, a single RealAudio clip can stream at many different speed. For the basics of SureStream, see "SureStream RealAudio and RealVideo" on page 35.

RealAudio Codecs

This section discusses the RealAudio codecs used by RealProducer. The codecs are listed in separate tables for voice, mono music, and stereo music. Voice codecs focus on the standard frequency range of the human voice. Music codecs have broader frequency responses to capture more high and low frequencies. The tables list each codec's optimum sampling rate and frequency response.

- Sampling rate

  Using a codec's optimum sampling rate in your audio source file ensures that the audio stays synchronized with other media in the presentation. It also prevents pitch shifting in audio resampling. Audio quality degrades if you use less than the optimum sampling rate. If you use a higher rate, it is best to use a multiple of the optimum rate. If the optimum rate is 8 kHz, for example, use a higher rate of 16 kHz or 32 kHz. When in doubt, use the CD-quality sampling rate of 44.1 kHz.

  > **Note:** RealProducer 8.5 and later can also take audio input sampled at a rate of 48 kHz, which is commonly used with digital video discs (DVD) and digital audio tape (DAT). RealProducer automatically resamples the input to the optimum rates for the selected RealAudio codecs.

- Frequency response

  Each codec has a particular frequency response measured in kilohertz (kHz). A codec with a higher frequency response reproduces a wider range of sound than a codec with a lower response. A measure of codec quality, the frequency response does not affect how you produce audio source files. RealAudio encoding always results in a clip of equal or lower quality than the original audio source. If the original audio source has an 8 kHz frequency response, encoding it with a codec that has a frequency response of 10 kHz produces a clip that still has a response of 8 kHz.

Voice Codecs

Voice codecs are for voice-only clips. The lowest-speed voice codec normally used to encode a RealAudio clip streams data at 16 Kbps. The lower-speed codecs (5, 6.5, and 8.5 Kbps) are used as SureStream duress streams that RealONE Player downshifts to if the connection bandwidth drops. They're also used to encode soundtracks for low-bandwidth RealVideo clips.

RealAudio Voice Codecs

RealAudio Codec	Sampling Rate	Frequency Response
5 Kbps Voice	8 kHz	4 kHz
6.5 Kbps Voice	8 kHz	4 kHz
8.5 Kbps Voice	8 kHz	4 kHz
16 Kbps Voice	16 kHz	8 kHz
32 Kbps Voice	22.05 kHz	11 kHz
64 Kbps Voice	44.1 kHz	20 kHz

Mono Music Codecs

As with the voice codecs, the lowest-speed mono music codec normally used with RealAudio streams data at 16 Kbps. The lower-speed codecs (6, 8, and 11 Kbps) are used as duress streams in SureStream clips, and to encode soundtracks for low-bandwidth RealVideo clips. When there are two versions of a codec, RealProducer uses the higher-response version by default.

RealAudio Mono Music Codecs

RealAudio Codec	Sampling Rate	Frequency Response
6 Kbps Music	8 kHz	3 kHz
8 Kbps Music	8 kHz	4 kHz
11 Kbps Music	11.025 kHz	5.5 kHz
16 Kbps Music	22.05 kHz	8 kHz
20 Kbps Music	22.05 kHz	10 kHz
20 Kbps Music—High Response	44.1 kHz	14.5 kHz
32 Kbps Music	44.1 kHz	14.5 kHz
32 Kbps Music—High Response	44.1 kHz	16 kHz
44 Kbps Music	44.1 kHz	20 kHz
64 Kbps Music	44.1 kHz	20 kHz

Stereo Music Codecs

RealPlayer G2 or later can play the stereo codecs listed in the following table. These stereo codecs do not stream slower than 20 Kbps because, if they did, they would not have enough frequency response for adequate sound.

RealAudio Stereo Music Codecs

RealAudio Codec	Sampling Rate	Frequency Response
20 Kbps Stereo Music	11.025 kHz	5 kHz
32 Kbps Stereo Music	22.05 kHz	8 kHz
44 Kbps Stereo Music	22.05 kHz	11 kHz
64 Kbps Stereo Music	44.1 kHz	16 kHz
96 Kbps Stereo Music	44.1 kHz	16 kHz

RealAudio 8 Stereo Music Codecs

RealAudio 8 introduces new stereo music codecs that greatly enhance sound quality, especially over high-speed connections. Only RealPlayer 8 and later can play RealAudio 8 clips. Users with RealPlayer G2 and RealPlayer 7 are prompted to upgrade to the latest version of RealONE Player before listening to a RealAudio 8 clip. When there are two versions of a codec, RealProducer uses the higher-response version by default. These codecs are also used to encode voice-with-music clips.

RealAudio 8 Stereo Music Codecs

RealAudio 8 Codec	Sampling Rate	Frequency Response
16 Kbps Stereo Music	22.05 kHz	4.3 kHz
20 Kbps Stereo Music	22.05 kHz	8.6 kHz
20 Kbps Stereo Music—High Response	22.05 kHz	9.9 kHz
32 Kbps Stereo Music	22.05 kHz	10.3 kHz
32 Kbps Stereo Music—High Response	44.1 kHz	13.8 kHz
44 Kbps Stereo Music	44.1 kHz	13.8 kHz
44 Kbps Stereo Music—High Response	44.1 kHz	16.0 kHz
64 Kbps Stereo Music	44.1 kHz	16.0 kHz
96 Kbps Stereo Music	44.1 kHz	16.0 kHz
105 Kbps Stereo Music	44.1 kHz	13.7 kHz
132 Kbps Stereo Music	44.1 kHz	16.5 kHz

(Table Page 1 of 2)

RealAudio 8 Stereo Music Codecs (continued)

RealAudio 8 Codec	Sampling Rate	Frequency Response
146 Kbps Stereo Music	44.1 kHz	16.5 kHz
176 Kbps Stereo Music	44.1 kHz	19.2 kHz
264 Kbps Stereo Music	44.1 kHz	22.0 kHz
352 Kbps Stereo Music	44.1 kHz	22.0 kHz

(Table Page 2 of 2)

Older RealAudio Codecs

The following tables list older RealAudio codecs that became obsolete with RealSystem G2. Some of these codecs are still used for compatibility with RealPlayer 5, though. RealPlayer G2 and later can play any clips encoded with these codecs. An "X" in the 5, 4, 3, 2, or 1 column indicates that a clip encoded with this codec can be played by that version of RealPlayer.

Older RealAudio Voice Codecs

RealAudio Codec	5	4	3	2	1	Sampling Rate	Frequency Response
8 Kbps Voice	X	X	X	X	X	8 kHz	4 kHz
15.2 Kbps Voice	X	X	X	X	–	8 kHz	4 kHz

Older RealAudio Mono Music Codecs

RealAudio Codec	5	4	3	2	1	Sampling Rate	Frequency Response
8 Kbps Music	X	X	–	–	–	8 kHz	4 kHz
12 Kbps Music	X	X	–	–	–	8 kHz	4 kHz
16 Kbps Music Low Response	X	X	X	–	–	8 kHz	4 kHz
16 Kbps Music Medium Response	X	X	X	–	–	11.025 kHz	4.7 kHz
16 Kbps Music High Response	X	X	X	–	–	11.025 kHz	5.5 kHz
32 Kbps Music	X	X	–	–	–	16 kHz	8 kHz
40 Kbps Music	X	X	X	–	–	22.05 kHz	11 kHz
80 Kbps Music	X	X	X	–	–	44.1 kHz	20 kHz

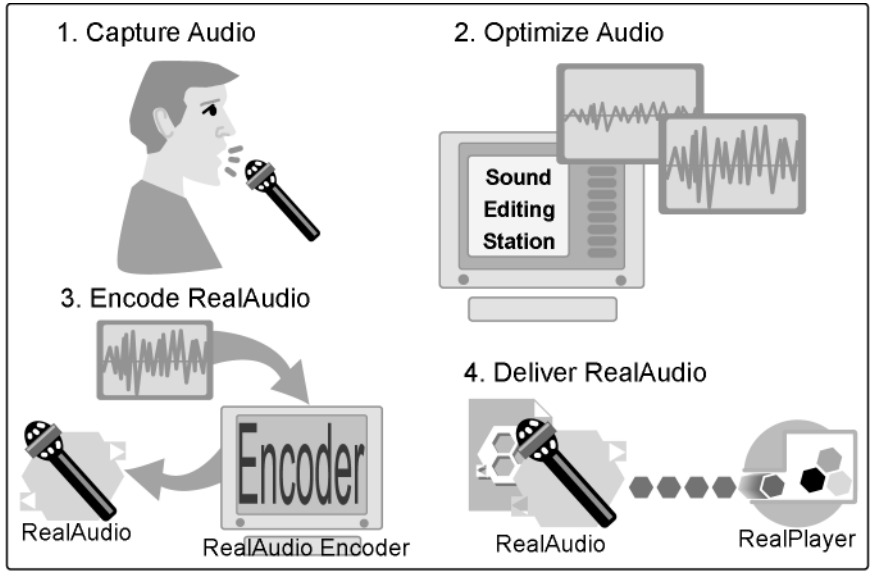
Older RealAudio Stereo Music Codecs

RealAudio Codec	5	4	3	2	1	Sampling Rate	Frequency Response
20 Kbps Stereo Music	X	X	X	–	–	8 kHz	4 kHz
32 Kbps Stereo Music	X	X	–	–	–	11.025 kHz	5.5 kHz
40 Kbps Stereo Music	X	X	X	–	–	16 kHz	8 kHz
80 Kbps Stereo Music	X	X	X	–	–	32 kHz	16 kHz

Steps for Streaming RealAudio

To produce a great RealAudio clip, you need to use great source material, high-quality equipment, and good production practices. This section surveys the steps involved in streaming a RealAudio clip.

Creating a RealAudio Clip



➤ To create a streaming RealAudio clip, follow these basic steps:

1. Capture audio source.

   You start audio production by capturing audio from a source, such as a person speaking into a microphone. You might also start with an audio source file from a compact disc, for example.

For More Information: "Capturing Audio" on page 50 provides guidelines for capturing audio.

2. Optimize the audio source.

   With the audio file digitized in a common file format such as WAV or AIFF, you can use a sound editor to optimize the audio file for encoding as a streaming clip. When broadcasting live, however, you encode audio input directly from the source, optimizing the audio during capture.

   For More Information: See "Optimizing Audio" on page 52 for tips on editing sound.

3. Encode the RealAudio clip.

   With your digitized file optimized or your live broadcast ready to go, you encode your source file in the RealAudio format. When you do this, you choose streaming bandwidths based on your target audiences.

   For More Information: "Encoding RealAudio with RealProducer" on page 54 provides encoding tips.

4. Deliver the RealAudio clip.

   When your presentation is ready to go, you make your RealAudio clip or broadcast available through your Web site. To combine a RealAudio clip with another streaming clip, such as a RealPix clip, you write a SMIL file.

   For More Information: Chapter 7 explains SMIL. See Chapter 20 for instructions on linking your Web page to a clip or a SMIL file.

Capturing Audio

A streaming clip reflects the quality of its audio source. Any quality problems within the source will affect the streaming clip as well. Because you cannot edit a broadcast, live Webcasting introduces several issues beyond those involved with delivering on-demand clips. This section will help you capture high-quality audio source files, or set up your sound equipment to deliver good broadcasts.

Source Media

If you plan to stream existing material, start with the best source possible. Use the cleanest recording with the least amount of unwanted noise. Compact discs (CDs) and digital audio tapes (DATs) are good source media, although well-recorded analog sources such as records, reel-to-reel tapes, and chrome (type II) cassettes can sound just as good. Try to avoid consumer-grade recording media such as Type I cassettes and VHS tapes.

Recording Equipment

Every piece of equipment in the audio chain—microphone, mixer, sound card, and so on—affects sound quality. If you intend to provide professional-quality audio content, invest in professional-quality audio equipment and software. Lesser equipment can add hiss and distortion, degrading sound clarity.

Shielded Cables

It is important to use high-quality, shielded cables. Using unshielded cables increases the likelihood of introducing line noise and radio frequency interference into recordings. Keep audio cables physically separated from power cords to minimize the introduction of noise. Also be sure to ground all equipment properly.

Input Levels

Setting correct input levels is crucial. All audio equipment has a signal-to-noise ratio, the ratio between the loudest possible sound the equipment can reproduce without distortion and its inherent "noise floor." Also called "clipping," distortion of this type is audible as a high-frequency crackling noise.

To get the best signal-to-noise ratio, set the input level on each audio device in the signal chain so that it uses its full range of available amplitude without distortion during the program's loudest sections. The signal chain typically includes a microphone, a mixing desk, a compressor, and a sound card. For each piece of equipment, set levels as close as possible to 0 decibels without going over that level.

Check for signal distortion at each point in the signal chain. Perform several test runs, and make sure that there are no peaks above maximum amplitude. Adjust the levels on your sound card mixer so that the input approaches but

does not exceed the maximum. Be conservative, though. Levels might suddenly increase if, for instance, an interviewee suddenly speaks loudly or a crowd at a sports event roars.

Volume Levels for Live Broadcasts

When broadcasting live audio streams, it is useful to have a dynamics compressor for gain compression (not data compression). This piece of audio equipment automatically adjusts the volume level. By providing a consistent volume level, it allows you to "set and forget" the input levels to RealProducer.

Sampling Rates

Try to capture sound with a sampling width of 16 bits. RealAudio codecs have different sampling rates that produce the best sound, however. If your sound card allows it, capture audio content at the optimum sampling rate for the codec you intend to use. The RealAudio encoder will convert the file to the optimum rate if necessary, but this is recommended only for static files. For live broadcasts, use a sound card that supports the optimum rate. This avoids the overhead entailed in converting the rate while encoding sound in real time.

> **For More Information:** "RealAudio Codecs" on page 45 lists the optimum sampling rates for each codec.

> **Tip:** You do not need to capture stereo sound if you plan to use a mono codec. However, many sound cards simply discard the right input channel in mono mode. If you have a mixing desk, pan all inputs to the center so that nothing is lost during the conversion to mono.

Optimizing Audio

If you are not broadcasting audio live, you work with digitized audio source files in supported formats such as WAV or AIFF. You then edit the audio files to optimize them. To do this, you need to be familiar with the features your editing program offers. This section gives you some optimization tips you can try with your editing software.

> **Tip:** Always keep copies of your audio source files. You cannot convert RealAudio clips back to their original source formats.

DC Offset

DC offset is low-frequency, inaudible noise that results from equipment grounding problems. If you don't remove it, it can skew the results of subsequent sound editing. Use your sound editor's **DC Offset** function immediately after recording a digital audio file.

> **Tip:** If your editing program has this option, remove DC offset during recording. This eliminates an editing step.

Normalization

Set sensible input levels when recording, and then use normalization to maximize the levels after recording. Your streaming files sound best when your digitized source has the highest possible gain without clipping. Digital audio files that do not use their full amplitude range produce low-quality streaming clips. If the amplitude range is too low, use your sound editor to adjust the range and increase the amplitude.

> **Tip:** Most sound editors have a **Normalize** function that maximizes levels automatically. Because some systems have trouble with files normalized to 100 percent, normalize to 95 percent of maximum, or to -0.5dB.

Dynamics Compression

Normalization maximizes the volume level of the audio file's loudest sections. Consequently, quiet sections may not encode as well. Dynamics compression evens out input levels by attenuating (turning down) the input when it rises above a specified threshold. Check your audio software for a **Compression** or **Dynamics** feature. You can control attenuation by specifying a compression ratio. This turns down the loudest sections, and you can readjust input levels accordingly.

> **Tip:** For multipurpose dynamics compression, set the threshold to -10dB, the ratio to 4:1, and the attack and release times to 100ms. Adjust the input level to get approximately 3dB of compression and an output level of about 0dB.

Equalization

Equalization (EQ) changes the tone of the incoming signal by "boosting" (turning up) or "cutting" (turning down) certain frequencies. Using EQ, you can emphasize certain frequencies and cut others that contain noise or unwanted sound. EQ can compensate for RealAudio codecs that do not have flat frequency responses (that is, codecs for which certain frequencies are not as loud after encoding). You can therefore use EQ to make a RealAudio clip sound as close as possible to the source recording.

> **Tip:** For voice-only content, you can make the file more intelligible by cutting frequencies below 100 Hz and carefully boosting frequencies in the 1 to 4 kHz range.

Encoding RealAudio with RealProducer

To produce RealAudio clips, you start with audio input from a live source, a digitized file, or media such as a tape or CD. You then encode a RealAudio clip from this input by using a RealAudio encoding tool. Some sound-editing programs can create RealAudio clips. But the most widely used tools for encoding RealAudio are RealProducer Basic (a free product) and RealProducer Plus, which are available from RealNetworks at:

http://www.realnetworks.com/products/producer/info.html

RealProducer on Macintosh accepts the formats widely used on that platform, whereas RealProducer on Windows and Unix supports the formats widely used on those platforms. Check the RealProducer manual for your operating system for a list of accepted formats, which may include:

- Audio Interchange Format (`.aiff`)
- Audio (`.au`)
- MPEG-1 (`.mpg`)
- QuickTime (`.mov`)
- Sound (`.snd`)
- WAV (`.wav`)

When you encode audio clips with RealProducer, you select the target audiences you want to reach, such as 28.8 Kbps modem users. You also indicate the audio type, whether voice or music. RealProducer then determines which RealAudio codecs are best to use. RealProducer uses

SureStream technology to encode the RealAudio clip so that it streams well for all your bandwidth choices, requiring only a few seconds of preroll when RealONE Player users request the clip.

> **For More Information:** Refer to the RealProducer user's guide or online help for step-by-step instructions for encoding RealAudio. A document explaining RealAudio codecs in-depth is at **http://service.real.com/help/library/blueprints.html**.
>
> **Note:** RealAudio encoding tools other than RealProducer may not include all the features described in the following sections.

RealAudio in Multiclip Presentations

If your RealAudio clip will be played along with another clip, you may need to change RealProducer's default codec selection by adjusting the RealAudio target audience settings. Encoding mono music for 28.8 Kbps modems usually entails a 20 Kbps codec, for example, leaving no bandwidth for the second clip. RealProducer has a set of multimedia defaults, though, that lower the clip's bandwidth. With these defaults turned on, RealProducer encodes the mono music clip at 8 Kbps, leaving 12 Kbps of bandwidth for the second clip.

Audio Compatibility with RealPlayer 5

RealProducer can include in SureStream clips a stream that RealPlayer 5 can play. This backward-compatible stream is at your lowest target bandwidth. For example, if your clip targets 28.8 Kbps modems, 56 Kbps modems, and DSL speeds, the RealPlayer 5 stream is for 28.8 Kbps modems. It uses an older RealAudio codec, and RealServer streams it to RealPlayer 5 using the older PNA protocol rather than the newer RTSP. RealPlayer 5 will receive this stream regardless of its connection speed.

CHAPTER

4

VIDEO PRODUCTION

RealNetworks introduced RealVideo with RealSystem 4, making streaming video available over the Internet. This chapter covers RealVideo production techniques, providing tips for capturing high-quality video, working with digitized video source files, and using RealProducer to encode your clips.

Understanding RealVideo

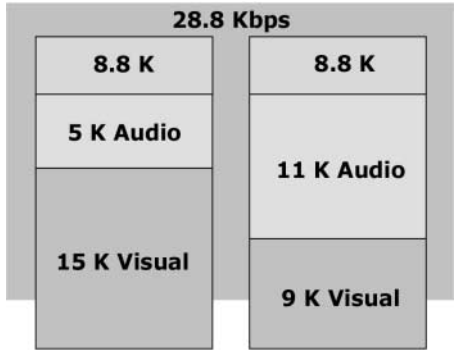
A video consists of two parts: the visual track and the soundtrack. In a RealVideo clip, the soundtrack is encoded with RealAudio codecs, and the visual track is encoded with a RealVideo codec. Both tracks are packaged in a RealVideo clip that, like a RealAudio clip, uses the file extension `.rm`. This section explains how RealVideo encodes a source video for streaming. This information will help you produce high-quality streaming clips.

> **Note:** Keep in mind that everything discussed about RealAudio clips in Chapter 3 also applies to the soundtracks in RealVideo clips.

RealVideo Bandwidth Characteristics

Because RealVideo uses RealAudio to encode a video's soundtrack, a chunk of the clip's bandwidth first goes toward the audio. The visual track is then squeezed into the bandwidth that's left. For 28.8 Kbps modems, for example, RealVideo clips stream at 20 Kbps, leaving 8.8 Kbps of modem bandwidth for overhead. How much bandwidth the visual track gets depends on how the audio is encoded. With a 5 Kbps RealAudio voice codec for the soundtrack, the visual track gets 15 Kbps. With an 11 Kbps music codec, though, the visual track gets just 9 Kbps.

Possible Audio and Visual Tracks in a 20 Kbps RealVideo Clip



At low bandwidths, how you encode the soundtrack can affect how the visual track looks. RealAudio music codecs typically consume more bandwidth than do voice codecs. Music's greater frequency range requires more data than does speech, so a music soundtrack consumes more bandwidth than a spoken one. A video with an audio narration might therefore look better than one accompanied by music, as there would be more bandwidth available for the visual track.

At higher streaming speeds, the soundtrack uses proportionally less of the clip's bandwidth, so differences in soundtrack encoding affect visual quality less. At speeds above 100 Kbps, you get high-quality sound that uses no more than a quarter of the clip's streaming bandwidth. The following table lists the standard RealVideo target audiences, giving the clip streaming speeds and the RealAudio codecs used for the soundtracks, broken out by audio type.

Audio Bit Rates for RealVideo Clips

		RealAudio Rate			
Target Audience	Clip Speed	Voice Only	Voice and Music	Mono Music	Stereo Music
28.8 Kbps modem	20 Kbps	6.5 Kbps	6.5 Kbps	8 Kbps	8 Kbps
56 Kbps modem	34 Kbps		8.5 Kbps		
64 Kbps single ISDN	45 Kbps	8.5 Kbps		11 Kbps	11 Kbps
112 Kbps dual ISDN	80 Kbps	16 Kbps	20 Kbps	16 Kbps	20 Kbps
Corporate LAN	150 Kbps	32 Kbps	32 Kbps	32 Kbps	32 Kbps

(Table Page 1 of 2)

Audio Bit Rates for RealVideo Clips (continued)

Target Audience	Clip Speed	RealAudio Rate: Voice Only	Voice and Music	Mono Music	Stereo Music
256 Kbps DSL/cable	225 Kbps	64 Kbps	44 Kbps	44 Kbps	44 Kbps
384 Kbps DSL/cable	350 Kbps		64 Kbps	64 Kbps	64 Kbps
512 Kbps DSL/cable	450 Kbps				96 Kbps

(Table Page 2 of 2)

Note: With SureStream technology, a single RealVideo clip can stream at many different speeds. For the basics of SureStream, see "SureStream RealAudio and RealVideo" on page 35.

RealVideo Frame Rates

Like RealAudio, RealVideo is "lossy," meaning that it throws out nonessential video data when encoding a clip. One way that RealVideo squeezes down clip sizes is by reducing the video's frame rate. The higher the frame rate, the smoother the motion:

- The standard frame rate for full-motion video is 24 to 30 frames per second (fps). At this speed, the human eye perceives movement as continuous—a phenomenon known as *persistence of vision*.
- A common rate for streaming video that approximates full-motion video is 15 fps. To most people, a 15 fps video flows smoothly, though not quite as fluidly as one at a higher rate.
- Below 15 fps, a video looks jerky.
- Below 7 fps, a video looks very jerky.
- Below 3 fps, a video essentially becomes a slideshow.

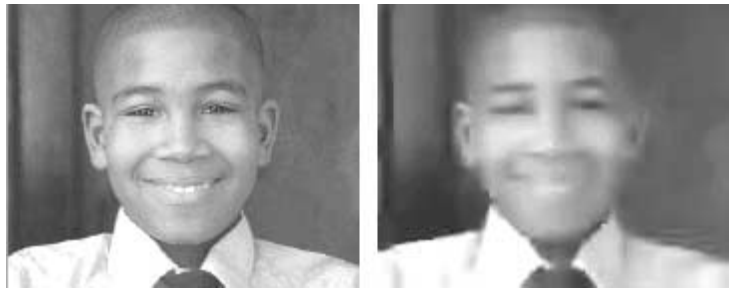
Most source videos start out at 15 to 30 fps. During encoding, RealVideo adjusts this frame rate downward as necessary, keeping the rate up in high-action scenes, reducing it in slow ones. Thus, your encoded clip will not have just one frame rate, but a mix of frame rates that varies with its content. If you follow good production practices, your clips will typically stream over slow- to medium-speed connections at 7 to 15 fps. At higher speeds, you'll get 15 to 30 fps. Many factors, though, affect a RealVideo clip's frame rate:

- The video's dimensions greatly affect frame rate. If you use too large of a window for your target bandwidth, you will not get a high frame rate. For more information, see "RealVideo Dimensions" on page 61.
- RealVideo 8 provides video quality superior to that produced by older RealVideo codecs. Using an older codec may result in a lower frame rate.
- Visually complex videos that show many objects moving across the screen simultaneously are hard to encode and may result in too low a frame rate.
- RealProducer gives you an option for smoother motion. Choosing this option raises the clip's overall frame rate and reduces visual quality, whereas choosing the option for better image quality lowers the frame rate.
- In a video that has a mix of fast and slow scenes, variable bit-rate encoding (VBR) and two-pass encoding generally help the fast scenes achieve a higher frame rate. See "RealVideo Options" on page 73 for more information.
- When encoding with RealProducer Plus, you can lower the bit rate of the RealAudio codecs used for a given clip. This gives more bandwidth to the visual track, helping to raise the frame rate.
- Scalable Video Technology (SVT) enables RealONE Player to lower the frame rate as necessary during playback to lighten a computer's CPU load. For more information, see "Scalable Video Technology" on page 64.

RealVideo Clarity

In addition to changing its frame rate, RealVideo can reduce a clip's streaming size by throwing out pixel data. A video stores information about each pixel in the frame. RealVideo, on the other hand, stores data for pixel groups. When bandwidth is tight, RealVideo shoehorns pixels with slightly different RGB values into the same group. These pixels then look identical rather than nearly identical. This may result in a loss of detail if compression is too high. The following illustration compares a smooth video with one that has lost detail through too much compression.

Smooth and Distorted Video



By using good production practices as described in this chapter, you can help keep the video's clarity intact during encoding. Also note the following points:

- The video's dimensions affect visual clarity. If you use too large of a window for your target bandwidth, visual clarity may suffer. For more information, see "RealVideo Dimensions" on page 61.
- When you encode with RealProducer, you can choose an option for better image quality. The video may be jerkier, though, because increasing the pixel data reduces the frame rate.
- A video with relatively stationary subjects ("talking heads") will have better visual quality than a video with rapid scene changes and a lot of movement.
- If you plan to launch a video in double- or full-screen mode as described in "Setting a Presentation's Starting Mode" on page 422, boost video clarity as much as possible during production and encoding. RealONE Player enlarges the clip by duplicating its pixels, which magnifies any defects.

RealVideo Dimensions

The following table lists four common RealVideo clip dimensions that maintain the 4:3 aspect ratio used in television. For each clip size, the table indicates the general playback quality you'll get when streaming a RealVideo 8 clip to various target audiences. "Excellent" video quality means few visual

artifacts and a frame rate that results in consistently smooth playback. Lower-quality video playback means more artifacts and a lower frame rate.

RealVideo Quality at Different Dimensions and Bandwidths

		Video Quality for Dimensions in Pixels			
Target Audience	Clip Speed	176 x 132	240 x 180	320 x 240	640 x 480
28.8 Kbps modem	20 Kbps	Excellent	Good	Fair	Poor
56 Kbps modem	34 Kbps				
64 Kbps single ISDN	45 Kbps			Good	
112 Kbps dual ISDN	80 Kbps		Excellent	Excellent	Fair
Corporate LAN	150 Kbps				Good
256 Kbps DSL/cable	225 Kbps				Excellent
384 Kbps DSL/cable	350 Kbps				
512 Kbps DSL/cable	450 Kbps				

The table shows that when streaming larger videos, you get good to excellent quality only at higher connection speeds. If you plan to stream clips over modems, you can first encode your clip at 320 by 240 pixels to test its quality. If you want better quality, shrink the video dimensions with your editing software or with RealProducer during encoding.

> **Tip:** When resizing a video with RealProducer, setting the RealVideo preferences to high-quality resize rather than a fast resize yields better results.

Different Clip Dimensions for Different Bandwidths

When you want to encode a single video for both low-speed and high-speed connections, you need to decide what video dimensions to use. Although with SureStream technology you can easily create a single RealVideo clip that streams at many bandwidths, using too large a clip results in low frame rates for viewers with dial-up modems. Therefore, you may want to use small video dimensions to reach the widest audience. High-speed viewers, however, will not receive the benefit of their hefty bandwidth.

You can resolve this dilemma by creating two or more versions of your video, each with different dimensions. You can use your video editing software to do this, or use RealProducer to resize or crop the video during encoding. Then encode each video for a few different streaming speeds. You can make the clips

available through separate links, or use a SMIL <switch> tag to let RealONE Player choose which version to play.

> **For More Information:** The basics of the <switch> tag are in "Understanding Switching" on page 363. For a sample <switch> tag, see "Switching for Different Video Sizes" on page 378.

RealVideo Codecs

RealVideo 8 is the standard RealVideo codec, but you can also encode with one of two RealVideo G2 codecs. The codec you use encodes all of a clip's SureStream streams. You cannot encode half the streams with the RealVideo 8 codec, for example, and the other half with a RealVideo G2 codec.

RealVideo 8 Codec

The RealVideo 8 codec results in visual quality markedly superior to that produced by the RealVideo G2 codecs. It requires more processing power, though, so encoding a clip with it takes longer than encoding the clip with a RealVideo G2 codec. RealPlayer 8 and later can play RealVideo 8 clips. Earlier versions of RealPlayer are prompted to autoupdate to the RealONE Player. RealNetworks recommends using this codec unless you need faster encoding performance during broadcasts, or you need to stream video to earlier versions of RealPlayer.

> **Note:** Most users upgrade their RealPlayers soon after a new version is released, so it is generally safe to use the RealVideo 8 codec.

RealVideo G2 Codecs

The RealVideo G2 codecs are older codecs used by RealProducer G2 and RealProducer 7. These codecs encode video clips faster than the RealVideo 8 codec does, but their visual quality is poorer. For a given streaming speed, using an older RealVideo codec rather than RealVideo 8 results in either a reduced frame rate or diminished visual quality at the same frame rate. Use one of the following RealVideo G2 codecs for faster encoding during broadcasts or if you need to stream video to versions of RealPlayer that cannot upgrade to Release 8:

- RealVideo G2 with SVT

  The RealVideo G2 with SVT codec is compatible with RealPlayer G2 Update 1 (version 6.0.6) or later. RealPlayers with lower version numbers are prompted to autoupdate to the latest version of RealONE Player before viewing the clip.

- RealVideo G2

  The RealVideo G2 codec without SVT is compatible with RealPlayer G2 and later. It is not compatible with RealPlayer 5 or earlier.

RealVideo Standard Codec

In RealProducer, you cannot explicitly choose the RealVideo standard codec, which was the default codec before RealSystem G2 was released. RealProducer uses this codec solely for backward compatibility with RealPlayer 5.

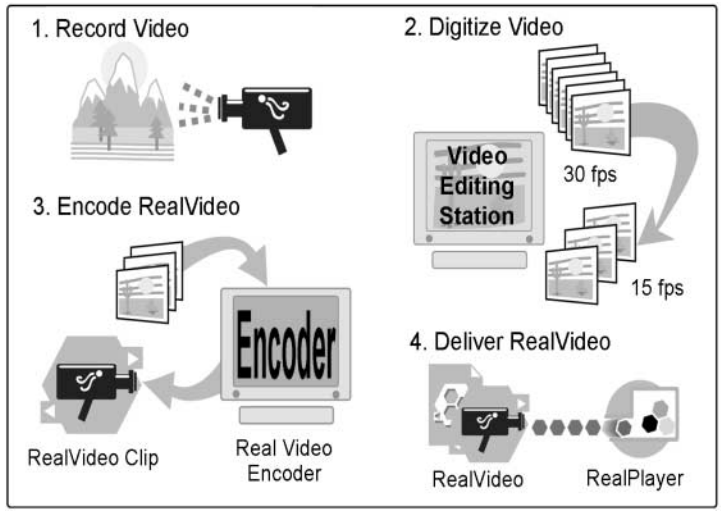
Scalable Video Technology

The RealVideo 8 and RealVideo G2 with SVT codecs include Scalable Video Technology (SVT), which scales down frame rates when clips play on slower computers. RealVideo's variable frame rate means that one scene may be encoded at 7 fps, whereas another one is encoded at 15 fps. High frame rates take a lot of processing power to decompress. Although fast PCs handle high frame rates well, slower PCs may have trouble. With SVT, RealONE Player can lower the frame rate "on the fly" to keep a PC's CPU from sputtering. So although a given scene is encoded at 15 fps, it may play on some RealONE Players at 8 fps if those computers lack the power to decompress 15 fps video clips.

Steps for Streaming RealVideo

When producing a RealVideo clip, you should choose the best source material and best equipment possible. The goal throughout the video production process is to get optimum video quality for each streaming speed.

Creating RealVideo Clips



➤ The following steps summarize how to create a RealVideo clip:

1. Capture the video content.

   To start video production, you capture the source video by shooting a scene with a video camera, for example, or gathering prerecorded content from a tape, satellite, laserdisc, or other source.

   > **For More Information:** "Recording Video" on page 66 provides guidelines for shooting a video.

2. Digitize and edit the video file.

   You next digitize the video to convert it to a standard file format, such as AVI or QuickTime. With your preferred video editing software, you can then edit the video as necessary. If you are broadcasting live, however, you encode the streaming video directly from the source.

   > **For More Information:** See "Digitizing Video" on page 67 for tips on video editing.

3. Encode the RealVideo clip.

   With your digitized file optimized or your live broadcast ready to go, you encode your source as a RealVideo clip. When you do this, you target a network bandwidth or a set of bandwidths.

For More Information: "Encoding RealVideo with RealProducer" on page 70 provides encoding tips.

4. Deliver the RealVideo clip.

   With your presentation ready to go, you make your RealVideo clip or broadcast available through your Web site. If you are combining video with another streaming clip, you write a SMIL file that assembles the pieces.

   For More Information: Chapter 7 explains how to create a SMIL file. See Chapter 20 for instructions on linking your Web page to a RealVideo clip or a SMIL file.

Recording Video

Read this section carefully if you intend to shoot a new video rather than use existing video content. Because video loses image quality if it's highly compressed, always start with the best video source available.

For More Information: For pointers on recording audio, see "Capturing Audio" on page 50.

Source Media Quality

Whether you shoot a video yourself or digitize existing material, start with a high-quality video media. The following are common video formats, listed in order of descending quality:

1. Betacam SP, also known simply as Beta. This format is common among video production professionals.
2. DV, miniDV, DVCam, or DVCPro.
3. Super-VHS (S-VHS) or HI-8mm.
4. VHS, 8mm.

Video Staging

Consider the video's final frame size before you shoot the first frame. Streaming over 28.8 Kbps modems requires a small video window, so you need to frame important visual elements well. For recommended clip dimensions, see "RealVideo Frame Rates" on page 59.

Scene Changes and Movement

The fewer things that change from frame to frame, the sharper the image will appear in a low-bandwidth video. You can do the following to cut down on unnecessary movement:

- Use a mounted camera rather than hand-held one. This greatly reduces the movement you inadvertently introduce into the scene when recording.
- Don't have a rapidly moving object fill the entire frame. But you don't want to pull the camera back too far either. You need to find a happy medium between close-ups and panoramic shots.

Of course, you don't want to eliminate all dynamic elements. When you do include rapid movement, allow enough time for objects to resolve. Because of low frame rates and high compression, objects coming to rest may appear blurry at first. If you have a dialog box popping up on a computer screen, for example, have the box remain stationary for a few seconds so that the image resolves.

> **Tip:** RealPix makes a great companion to RealVideo. When presenting a lecture, for example, use RealVideo to show the speaker, and use a RealPix slideshow to present visual aids such as information written on a blackboard. For more on RealPix, see "Images" on page 28.

Colors and Lighting

Bright lighting at a constant exposure keeps the foreground detail crisp. Use uniformly dark colors for backgrounds, and uniformly light colors (but not whites) for clothing. Complex textures such as paisley and stripes degrade the final image quality with unwanted visual effects.

Digitizing Video

If you are not broadcasting RealVideo streams live, you digitize the source video on your computer or video editing station. You can then edit the file with your preferred editing software before encoding it as a RealVideo clip. This section provides tips for digitizing video.

> **Tip:** Always keep copies of the video source files. You cannot convert RealVideo clips back to their original source formats or any other streaming formats.

S-Video Output

Video playback devices commonly have at least two common output types—S-video and composite. Use S-video, as it produces better results.

Color Depth

Always use 24-bit color. Lower color resolution results in poor clips.

Digitized Video Formats

It is better to work with uncompressed formats. Otherwise, you compress the source once when you digitize it and again when you encode it as RealVideo. This double compression can decrease the image quality. Use a compressed source format only if your RealVideo encoding tool supports the file as input. You can use compressed AVI files as long as the computer used to encode RealVideo clips has the same Video for Windows driver used to compress the AVI file.

Video Capture Frame Rates

Capture the video content at 15 frames per second (fps) if your clip will stream at less than 100 Kbps. For higher-speed streaming, capture it at 30 fps. Because a RealVideo clip's frame rate varies, some scenes may be encoded at less than the capture rate depending on bandwidth constraints. SVT enables RealONE Player to lower the frame rate on computers that have low processing power, ensuring that high frame rates will not overburden slow computers.

For More Information: For more information on SVT, see "Scalable Video Technology" on page 64.

Video Capture Screen Sizes

Unless you are short on disk space or your video capture card recommends different dimensions, capture video at 320 pixels wide by 240 pixels high. You can capture full-motion video at the full-screen size of 640 by 480 pixels if all of the following are true:

- Your clips will stream at speeds higher than 100 Kbps.
- Your encoded clips will be larger than 320 by 240 pixels.

- You have a video workstation capable of digitizing full-motion, full-screen video. Standard PCs typically cannot handle this large a load.

Using your editing software or RealProducer, you can reduce the RealVideo clip dimensions if necessary. For recommended clip sizes, see "RealVideo Dimensions" on page 61.

Computer Speed and Disk Space

Because video capture places a large burden on a computer's CPU and hard drive, use the fastest computer you have available. To avoid dropping frames during video capture, use a hard drive specially made for audio and video work. On Windows computers, you can use any video capture card that supports Video for Windows.

Use the following formula to calculate the approximate size in megabytes of a digitized video file:

$$\frac{\text{(pixel width) x (pixel height) x (color bit depth) x (fps) x (duration in seconds)}}{\text{8,000,000}}$$

Suppose you want to capture a three-minute video at 15 frames per second, with 24-bit color, in a window that is 320 by 240 pixels. As you can see from the following equation, your digitized source file would be approximately 622 MB:

```
(320) x (240) x (24) x (15) x (180) / 8,000,000 = 622 Megabytes
```

If necessary, you can conserve disk space by decreasing the clip dimensions or lowering the frame rate, or both.

Video Source File Size Limit

The Macintosh and some Windows file systems limit a single file to 2 GB (2048 MB) in size. At a 320-by-240 size and 15 fps, this translates to about 9.5 minutes of video. Certain video production programs support the OpenDML (AVI 2.0) standard, which allows the creation of files larger than 2 GB. RealProducer may be able to accept a video source file larger than 2 GB as input, depending on the production software used to create the file.

If you plan to produce long videos or videos with large dimensions, check whether or not your video production software is limited to a 2 GB output file size. If it is not limited to 2 GB, create a video file larger than 2 GB and test to determine if RealProducer can accept the file as input.

For More Information: Search RealNetworks' Knowledge Base for "AVI limit" at **http://service.real.com/kb/index.html**.

Tip: If you are limited to 2 GB for the video source file and you need to produce a larger video, you can create separate video source files (each 2 GB or smaller) and encode them as separate RealVideo clips. Then, merge the clips using RealProducer's editing tools. See the *RealProducer User's Guide* for more information.

Encoding RealVideo with RealProducer

To produce RealVideo clips, you start with video input from a live source, a digitized file, or media such as a tape or CD. You then encode a RealVideo clip from this input using a RealVideo encoding tool. The most widely used tools for encoding RealVideo are RealProducer Basic (a free product) and RealProducer Plus, which are available from RealNetworks at:

http://www.realnetworks.com/products/producer/info.html

RealProducer on Macintosh accepts the input formats widely used on that platform, whereas RealProducer on Windows or Unix supports the formats widely used on those platforms. Check the RealProducer manual for your operating system for a list of accepted formats, which may include:

- AVI (`.avi`), uncompressed (recommended) or compressed
- MPEG-1 (`.mpg`)
- QuickTime (`.mov`), uncompressed (recommended) or compressed

When you encode RealVideo clips, you choose a bandwidth target or set of targets, then set parameters such as audio type (voice or music) and an emphasis on either smooth motion or sharp images. RealProducer then decides which RealAudio codecs are best to use for the soundtrack. It uses SureStream technology to encode the RealVideo clip for all your bandwidth choices, ensuring that each stream requires only a few seconds of preroll.

For More Information: Refer to the RealProducer manual or online help for step-by-step instructions for encoding clips. A document explaining RealVideo codecs and options in-depth is at **http://service.real.com/help/library/blueprints.html**.

Note: RealVideo encoding tools other than RealProducer may not include all the features described in the following sections.

RealVideo Streaming Speeds

For all practical purposes, RealVideo's streaming bit rate is infinitely scalable. You can encode RealVideo at any speed you want, from 20 Kbps to hundreds or thousands of kilobits per second. Plus you can encode precisely at any bandwidth you choose, such as 89 Kbps, 117 Kbps, 575 Kbps, 1.5 Mbps, and so on. With RealProducer Plus, you do this by adjusting RealVideo's target audience settings.

If your RealVideo clip will play in parallel with another clip, you may need to lower RealVideo's bit rate for each target audience. This way, the clip does not consume all of a connection's bandwidth. For a 56 Kbps modem, for example, RealProducer standardly encodes the clip to stream at 34 Kbps. Using the RealVideo target audience settings, you can lower this value to 20 Kbps, for example, leaving 14 Kbps free for another clip.

Whenever you lower RealVideo speed, the new bit rate includes the rate of the RealAudio codec used for the soundtrack. If you use a 32 Kbps RealAudio codec and set a total RealVideo bit rate of 36 Kbps, you will not have much bandwidth left for the visuals (maybe a frame every few seconds). You'll need to select a lower-bandwidth RealAudio codec.

There are no strict rules for the ratio of soundtrack bandwidth to total RealVideo bandwidth. The ratio depends entirely on whether you want to emphasize the audio track or the visuals. RealVideo normally allots no more than 25 percent of the total bandwidth to the soundtrack because the visual information contains much more data. At bandwidths of several hundred kilobits per second, you can use considerably less bandwidth for the soundtrack in relation to the visual track.

Video Compatibility with RealPlayer 5

RealProducer can include in SureStream clips a stream that RealPlayer 5 can play. This backward-compatible stream is set for your lowest target bandwidth. For example, if your clip targets 28.8 Kbps modems, 56 Kbps modems, and DSL speeds, the RealPlayer 5 stream is for 28.8 Kbps modems. It uses older RealAudio and RealVideo codecs, and RealServer streams it to RealPlayer 5 using the older PNA protocol rather than the newer RTSP.

RealVideo Filters

RealProducer's preferences include filtering options you can use when encoding RealVideo clips. These filters remove artifacts that appear in the encoded clips because of the methods used to create the source video.

Noise Filters

Noise distortion is similar to the "snow" that often shows up on TV screens as a result of signals received over an antenna. RealProducer has low and high noise filters you can use if your source video clip has a little noise or a lot of noise, respectively. The best way to eliminate noise, though, is to follow good production practices, as described in "Recording Video" on page 66.

Resizing Filters

When you resize a RealVideo clip with RealProducer, you can use a resizing filter that performs either a fast resize or a high-quality resize. These resizing options affect the source video only when you make the clip smaller. They tell RealProducer to throw out video data by using either a quick method (fast resize) or a complex analysis (high-quality resize). A high-quality resize results in a superior image, but it also lengthens the encoding time.

Inverse-Telecine Filter

The Inverse-Telecine filter is for cinematic film that was transferred to NTSC video. It has no effect on PAL video. Film is usually photographed at 24 frames per second (fps), and the film-to-video conversion (called "telecine") duplicates some frames to achieve the NTSC standard of 30 fps. To strip out redundant frames, use the inverse-telecine filter when encoding NTSC video that was transferred from film and has a frame rate of 25 to 30 fps.

De-Interlace Filter

The de-interlace filter removes jaggedness from interlaced video. It is useful only for interlaced source video that is at least 240 lines high. Typical source video used for television is 480 lines high. If you digitize the video with a video capture card that captures at 240 lines or fewer, the card throws out either the odd or the even lines, de-interlacing the video itself.

RealVideo Options

The RealProducer preferences include several RealVideo encoding options that affect the quality of RealVideo clips by modifying RealProducer's encoding methods.

Variable Bit Rate Encoding

Variable bit rate (VBR) encoding is optional for RealVideo. It varies a clip's *playback* bit rate while keeping the *streaming* bit rate constant. This gives more bandwidth to scenes that are hard to compress and less to scenes that are easy to compress. In a video that streams at 225 Kbps, for example, one second may get 150 Kb of data while another gets 300 Kb. VBR encoding generally provides video quality superior to that achieved with constant bit rate (CBR) encoding, making the most difference in videos with a mix of high- and low-action scenes.

Two-Pass Encoding

With two-pass encoding, RealProducer runs through the source video once to gather information. It then makes a second pass to encode the RealVideo streams. Two-pass encoding helps the most with variable bit rate (VBR) encoding. When you use two-pass encoding with VBR, RealProducer analyzes the entire source video first to determine how best to vary the playback bit rate. When you don't use it, RealProducer analyzes small sections of the source video during encoding, creating a string of VBR sections within the clip.

Loss Protection

RealProducer's loss-protection feature adds error-correction data to RealVideo streams, helping them maintain quality in lossy environments. RealProducer adds as much error-correction data as it can without lessening the video quality. Although you'll get more benefit from this feature when streaming across the Internet rather than over an intranet, it is useful to turn loss protection on for all encoded content.

CHAPTER

5

FLASH ANIMATION

Macromedia Flash makes it easy to put animation on the World Wide Web. Streamed through RealSystem, Flash clips can deliver visually arresting animations and e-commerce applications to RealONE Player. This chapter provides guidelines for creating and optimizing Flash clips that stream to RealONE Player. For instructions on developing Flash animation, refer to the Flash user's guide.

> **For More Information:** Learn more about Flash from Macromedia's Web site at **http://www.macromedia.com/software/flash**.

Understanding Flash

Flash is well-suited for linear presentations that have a continuous audio track and animated images synchronized along a timeline. Such presentations could include:

- demonstrations, training courses, and product overviews
- full-length cartoons for entertainment and education
- product advertisements
- movie trailers
- Karaoke

With Flash commands, you can build interactive icons and forms for:

- electronic commerce
- on-screen navigation
- Internet radio tuners
- e-mail registration

This section explains how Flash works with RealSystem. This knowledge will help you produce high-quality streaming animation.

RealSystem Software Versions for Flash

Streaming Flash version 3 or 4 to RealONE Player requires RealServer 8 or later. Earlier versions of RealServer stream only Flash 2. RealPlayer 8 or RealONE Player is required to play Flash 3 or 4 clips. RealPlayer G2 or 7 will autoupdate to the latest RealONE Player release when it encounters a Flash 3 or 4 clip.

RealONE Player does not support the Flash 5 Player format. You can develop your animation with the Flash 5 program, but your exported Flash Player clip must be in the Flash 2, 3, or 4 format. Note that the Flash 5 program can automatically export and tune your clips in the Flash 4 format for streaming to RealONE Player.

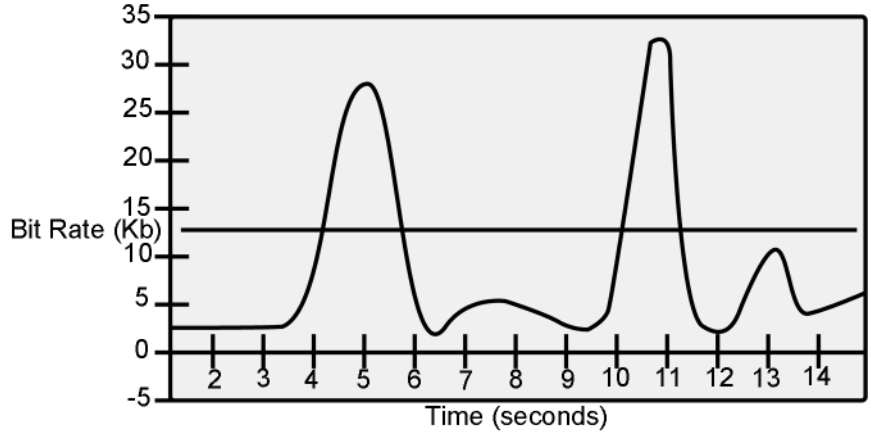
Flash Bandwidth Characteristics

As with any streaming clip, you develop a Flash clip with a target audience bandwidth in mind. The table "Maximum Streaming Rates" on page 32 lists the highest rate at which your Flash clip should stream for various network connection speeds. Keep in mind, too, that if your Flash clip streams along with other clips, the combined streaming speed of all the clips should not exceed the maximum speed for the target audience. This helps ensure that your presentation does not rebuffer frequently.

Because most Internet users have 28.8 or 56 Kbps modems, RealNetworks recommends that you target dial-up modem audiences. Fortunately, Flash clips streamed over a 28.8 Kbps modems can have a visual impact comparable to that of a video streaming at a significantly higher bit rate. This is because Flash clips transmit vector information rendered by the viewers' computers. Hence, the quality of Flash animation depends more on a computer's CPU and graphics capabilities than on the amount of streamed data.

Because it is vector-based, Flash does not consume bandwidth evenly. When a scene starts, for example, its groups and symbols are streamed, requiring a lot of data transfer. After that, only lightweight instructions for manipulating groups and symbols are needed. This following illustration shows a Flash clip that targets a streaming speed of 12 Kbps. At 5 and 10 seconds into the clip's timeline, bandwidth use spikes because the clip needs more than 12 kilobits of data to change scenes or to introduce new objects in a key frame.

Sample Bit Rate Requirement for a Streaming Flash Clip



If it encounters spikes, RealONE Player buffers the data, delaying playback until all of the necessary data has arrived. For your clip to stream well, you must eliminate spikes by tuning the finished clip. Tuning the clip also sets the clip's streaming bit rate and preroll. The Flash 5 program can export and tune a clip automatically. Or, you can tune an exported clip manually with the Flash tuner included in the RealSystem Authoring Kit, which you can download from the following Web site:

http://www.realnetworks.com/products/authkit/index.html

> **Tip:** You will not know how well your clip streams until you tune it. Because you may need to revise the animation to make the clip stream well for your chosen audience, export and tune the animation frequently as you develop it.

Flash Clip Size

Tuning your Flash clip guarantees that it streams at your chosen bit rate. If your animation is too complex, however, tuning it to a low bit rate may cause an unacceptably high preroll in RealONE Player. The best way to guarantee a low preroll is to keep the ratio of clip size to clip length low. The following are tips for keeping the Flash clip size as small as possible as you develop your animation:

- Reduce key frames.

  Excessive key frame changes increase bandwidth consumption. Minimize the number of key frames and simplify the objects within key frames.

- Use symbols instead of groups.

  Flash stores a symbol once and can then refer to it at any time, with each reference adding little to the file size. However, it stores a group definition each time the group is used. Using a group three times, for example, stores the same data in the file three times. Using symbols instead of groups can therefore reduce file size significantly.

- Simplify elements.

  Simplify the elements that you draw or import into Flash. Under **Modify>Curves**, use the **Smooth** and **Straighten** commands on lines and curves to strip away unneeded point and path information. This reduces the amount of data stored for each element. Use **Optimize** to optimize the data reduction while maintaining acceptable screen appearance. Because screen resolution is lower than print resolution, you can eliminate minute details without compromising appearance.

- Do not include event sound.

  As described in "Adding Audio to Flash" on page 79, RealONE Player does not support event sounds. Including event sounds in the exported Flash Player clip wastes bandwidth. Either remove event sounds from the source Flash file (.fla) or export the Flash Player clip (.swf) without sound.

- Adjust JPEG quality when exporting.

  If your animation has imported graphics, set the JPEG quality to no greater than 50—possibly as low as 30—when exporting the .fla file to a .swf clip.

Flash CPU Use

Bandwidth use is not the only consideration when developing Flash animation. Because it is vector-based, Flash performs complex calculations on the user's computer to display the animation. Operations that require many calculations on top of the computer's normal load may adversely affect playback. Newer computers typically have processors that are fast enough to handle Flash and other clips streamed in parallel, but older computers may not have this capacity. To support the widest audience possible, follow these recommendations to reduce Flash CPU requirements:

- Reduce the frame rate.

  Macromedia recommends a Flash frame rate of 12 frames per second (fps). If you combine a Flash clip with another clip that needs considerable processing power, though, you may need to lower this frame rate to accommodate slow computers. Try 9 fps or 7 fps when combining Flash with RealAudio, for example. These rates provide acceptably smooth motion without overburdening most processors.

- Optimize tweening.

  The tweening process interpolates the motion between key frames. Interpolating multiple objects and color effects at the same time will adversely affect playback. Other actions related to tweening that slow down playback include changing large areas of the screen between frames and using gradient fills.

- Decrease the number and size of objects moving simultaneously.

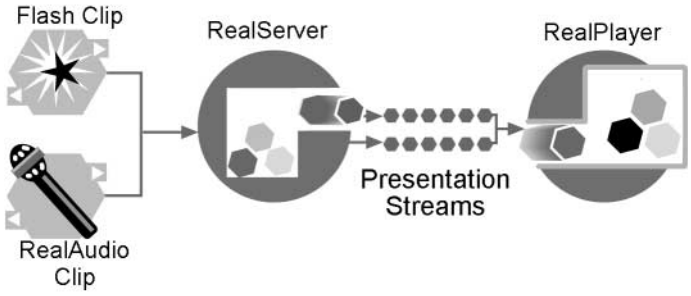
  RealONE Player must redraw areas where action occurs, thus consuming CPU power. To minimize this, localize tweening to a small portion of the screen so that the entire screen does not have to be redrawn. This way, file size remains the same, but only one part of the screen is redrawn.

Adding Audio to Flash

RealONE Player does not support audio embedded in a Flash clip, so it will not play event sounds for mouseovers and button clicks, for example. You can include sound for a Flash clip only by exporting the soundtrack and encoding it as a RealAudio clip that plays in parallel with Flash. Therefore, audio tracks are beneficial primarily for linear clips such as a cartoons, rather than for interactive applications.

> **Tip:** Flash provides different methods for incorporating sound into an animation. Use the stream synchronization setting.

A Flash Soundtrack Uses a Separate RealAudio Clip



To create separate animation and audio clips, you first synchronize your animation with an imported sound file, such as a WAV or AIFF file. You then export two files:

- A Flash Player clip that contains no sound.
- A soundtrack that you encode as a RealAudio clip.

Using SMIL, you synchronize both clips for streaming. The Flash 5 program can create a SMIL file and export your soundtrack as a RealAudio clip automatically. If you use an earlier version of Flash, you need to export the soundtrack manually, encode it as a RealAudio clip, and write the SMIL file. "Streaming a Flash Clip" on page 88 summarizes this exporting process.

Bandwidth Division Between Flash and RealAudio

When you export and encode your Flash soundtrack as a SureStream RealAudio clip, all viewers get the same Flash clip, but they get different RealAudio streams depending on their network connection speeds. For any network connection, determining your Flash and RealAudio clip speeds is a two-step process:

1. Decide which RealAudio codecs to use to encode the soundtrack. All codecs are listed in "RealAudio Codecs" on page 45.
2. For your lowest-speed target audience, subtract the lowest RealAudio streaming speed from the target's maximum streaming speed to get the Flash clip's maximum streaming speed.

   For More Information: The table "Maximum Streaming Rates" on page 32 lists the streaming speeds for various network connection speeds.

Targeting 28.8 Kbps Modems

The following table lists possible RealAudio and Flash bit-rate combinations for 28.8 Kbps modems, which have a maximum streaming speed of 20 Kbps. If you choose an 8 Kbps music codec for RealAudio, for example, you have 12 Kbps of streaming bandwidth left for Flash.

Bandwidth Divisions between RealAudio and Flash at 20 Kbps

Soundtrack Type	RealAudio Codec	Flash Maximum Speed
Voice	5 Kbps Voice	15 Kbps
	6.5 Kbps Voice	13.5 Kbps
	8.5 Kbps Voice	11.5 Kbps
Music	6 Kbps Music	14 Kbps
	8 Kbps Music	12 Kbps
	11 Kbps Music	9 Kbps

Targeting 56 Kbps Modems

Suppose you want to reach 56 Kbps modems, which have a maximum streaming speed of 34 Kbps. The following table lists some RealAudio codecs you can use, indicating for each codec the streaming speed left for the Flash clip.

Bandwidth Divisions between RealAudio and Flash at 34 Kbps

Soundtrack Type	RealAudio Codec	Flash Maximum Speed
Voice	6.5 Kbps Voice	27.5 Kbps
	8.5 Kbps Voice	25.5 Kbps
	16 Kbps Voice	18 Kbps
Music	11 Kbps Music	23 Kbps
	16 Kbps Music	18 Kbps
	20 Kbps Music	14 Kbps

Targeting Both 28.8 and 56 Kbps Modems

To target both 28.8 and 56 Kbps modems, decide first how to reach the 28.8 Kbps audience. For a voice soundtrack, for example, you might use a 6.5 Kbps RealAudio voice codec, leaving 13.5 Kbps for Flash. To reach 56 Kbps modems, you would encode the soundtrack as a SureStream RealAudio clip using both the 6.5 Kbps voice codec and a 16 Kbps voice codec. Users with 56 Kbps

modems then get 16 Kbps of RealAudio data along with the 13.5 Kbps Flash clip. This puts the streaming speed for this combination at 29.5 Kbps, a little less than the 34 Kbps maximum.

Tips for Choosing RealAudio Codecs

Here are some tips for selecting a RealAudio codec to use with a streaming Flash clip:

- If sound quality takes precedence, use the fastest RealAudio codec that still leaves enough bandwidth for acceptable animation.
- When animation is complex, use low-speed RealAudio codecs targeted for voice. This increases the bandwidth available for the animation.
- If possible, do not select the lowest-speed RealAudio codec. SureStream clips include a duress stream that is used if the connection bandwidth falls. An 8 Kbps music clip, for example, includes a 6 Kbps duress stream. If you encode the clip using just the 6 Kbps codec, RealONE Player will have no duress stream to fall back on.
- To encode a RealAudio clip with exactly the codec you want, you may need to change the RealAudio default target audience settings. You need RealProducer Plus to do this, because RealProducer Basic does not allow changes to the default settings.

Using Interactive Flash Commands

Because RealONE Player supports all Flash 3 and Flash 4 commands, you can make your presentations interactive by adding buttons and forms. In some cases, Flash commands work differently in RealONE Player than in the Flash browser plug-in. This section provides guidelines for using Flash commands with RealONE Player presentations.

Flash Clip Timeline Commands

Flash has several commands you can use to control the Flash clip's timeline. In a Flash 3 or 4 clip, these commands affect only the Flash clip. The presentation and all other clips playing along with the Flash clip continue

through their timelines normally. In a Flash 2 clip, these commands affect all clips playing in RealONE Player.

Interactive Flash Commands

Command	Function
`Play`	Begins or resumes Flash clip playback.
`Stop`	Pauses Flash clip until a `Play` command is issued. With a Flash 3 or 4 clip, all other clips play normally. With a Flash 2 clip, all other clips pause.
`Go To and Stop`	Seeks to the designated frame in the Flash clip and pauses. The Flash clip timeline resumes on a `Play` command. With a Flash 3 or 4 clip, all other clips play normally. With a Flash 2 clip, all other clips seek to the same point in the presentation timeline and then pause. See also "Go To Commands" on page 84.
`Go To and Play`	Seeks to the designated frame in the Flash clip, buffers the clip preroll, and begins playback. With a Flash 3 or 4 clip, all other clips play normally. With a Flash 2 clip, all other clips seek to the same point in the presentation timeline and then resume playback.
`Get URL`	Sends the URL to the browser. Because the user has to return to the animation manually, use this only at the end of a clip. Also note that a SMIL file can define clickable hyperlinks that overlay a Flash clip. See Chapter 14 beginning on page 279.

RealONE Player Commands

As noted in the preceding table, commands such as **Play**, **Stop**, and **Go To** in Flash 3 and Flash 4 clips affect only the Flash clip. Using Flash's **Get URL** command, though, you can play, stop, or pause all clips playing in RealONE Player. You can also launch a URL in a new RealONE Player window. You do this by sending RealONE Player a command (rather than a URL) through **Get URL**.

Seeking Into a Presentation

The following value for **Get URL** instructs RealONE Player to seek to the specified time in the presentation timeline:

```
command:seek(time)
```

For example, the following command instructs RealONE Player to seek to 1:35.4 in the presentation timeline:

```
command:seek(1:35.4)
```

The time format is as follows:

```
dd:hh:mm:ss.xyz
```

Here, `dd` is days, `hh` is hours, `mm` is minutes, `ss` is seconds, `x` is tenths of seconds, `y` is hundredths of seconds, and `z` is milliseconds. Only the `ss` field is required. When the time value does not include a decimal point, RealONE Player reads the last field as the seconds. For example, `1:30` means 1 minute and 30 seconds, whereas `1:30:00` means 1 hour and 30 minutes. Note that all of the following commands are equivalent. Each seeks to the point 90 minutes into the presentation timeline:

```
command:seek(1:30:00.0)
command:seek(90:00)
command:seek(5400)
```

Playing, Pausing, or Stopping a Presentation

The following values for **Get URL** cause RealONE Player to play, pause, or stop the presentation, respectively:

```
command:play()
command:pause()
command:stop()
```

Popping Up New Media Windows

Using the **Get URL** command, you can open streaming presentations in new RealONE Player media windows. You can open as many player windows as the computer's CPU and memory allow. For information on opening a new media window, see "Opening a New Media Window Through RealText or Flash" on page 306.

Go To Commands

Use **Go To** commands only when adding interactivity to a Flash clip. Do not use them to advance from one scene to the next. When you export your animation in the Flash Player format, scenes are concatenated so that the animation flows from one scene to the next. A **Go To** command causes RealONE Player to seek to the target frame. If RealServer has not yet streamed the target frame, RealONE Player halts clip playback, issues a seek request to RealServer, and rebuffers the new data as it comes in.

When you use at least one **Go To** command in a Flash 3 or 4 clip, RealONE Player caches the entire clip in memory. It assumes that the clip is interactive

and that the **Go To** commands are meant to move the viewer from one part of the clip to another based on input such as a button click. After RealServer has streamed the frames containing the **Go To** command and its target, RealONE Player does not need to rebuffer the clip when the viewer gives the command.

Using a **Go To** command raises RealONE Player's memory requirement for playing the clip. This is generally not a problem, because Flash memory requirements are low. When authoring long, linear animations, though, avoid using **Go To** commands whenever possible. When no **Go To** commands are present, RealONE Player discards clip data it no longer needs. This helps guarantee good-quality playback on computers that are low on available memory.

Load Movie Commands

RealONE Player imposes a restriction on using Flash's **Load Movie** command to import a second Flash clip into a clip that is playing. If the clips use the RTSP protocol, RealServer stops the first clip and streams the second clip as a new RealONE Player presentation rather than streaming the second clip as part of the initial presentation. The **Load Movie** command works properly only when clips are downloaded with HTTP. There are two ways to manage this:

- Stream the first clip with RTSP by using `rtsp://` in the SMIL or Ram file URL for the clip. In a **Load Movie** command, use a fully qualified HTTP URL for the clip. RealONE Player will then request the clip with the given URL. This is the preferred solution because the first clip uses RTSP, which is a better protocol for streaming.

  > **Tip:** RealServer supports both RTSP and HTTP. You can therefore put all clips in the same RealServer directory, streaming the first one with RTSP and all of the others with HTTP. Just be sure not to include `/ramgen/` in the URLs used with the **Load Movie** command.

- The second solution is to download all clips by using HTTP. Use `http://` in the SMIL or Ram file URL to the initial clip. In a **Load Movie** command, you can then refer to an imported clip using just its file name. RealONE Player requests subsequent clips using the same HTTP URL (except for the different file names) used to download the first clip.

  > **Tip:** If your presentation does not use SMIL, use a Ram file instead of Ramgen to list the HTTP URL to the first clip.

RealServer's Ramgen utility adds /ramgen/ to the first clip's URL. When RealONE Player reuses this URL, the /ramgen/ component starts a new presentation.

Timeline Slider Activity with Multiple Clips

If your presentation includes multiple Flash clips integrated with **Load Movie** commands, the RealONE Player slider reflects only the first clip's timeline. Suppose that a clip plays for five minutes and then loads another clip. The RealONE Player slider is active only for the five minutes the first clip plays. After that, the second clip plays normally, but RealONE Player indicates that the presentation has finished by resetting the timeline slider and disabling the stop button. Viewers can still perform interactive functions and stop the second clip by using Flash's contextual menu, though.

Using SMIL Instead of Load Movie

You need to use the **Load Movie** command to insert a new Flash clip into a Flash clip that is already playing. You do not need to use this command to play two or more Flash clips in sequence, though. Instead, you can use SMIL to define the sequence. This overcomes the URL and timeline limitations described above. To play two clips in sequence, for example, you write a SMIL file that looks like the following:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <body>
    <seq>
      <animation src="rtsp://realserver.example.com:554/media/cartoon1.swf"/>
      <animation src="rtsp://realserver.example.com:554/media/cartoon2.swf"/>
    </seq>
  </body>
</smil>
```

You can also use SMIL to combine each Flash clip with a RealAudio clip. The example below has two clip groups that play in sequence. Each clip group is composed of a Flash clip and a RealAudio clip played in parallel:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <body>
    <seq>
      <par>
        <animation src="rtsp://realserver.example.com:554/media/cartoon1.swf"/>
        <audio src="rtsp://realserver.example.com:554/media/sound1.rm"/>
      </par>
      <par>
```

```
      <animation src="rtsp://realserver.example.com:554/media/cartoon2.swf"/>
      <audio src="rtsp://realserver.example.com:554/media/sound2.rm"/>
    </par>
  </seq>
 </body>
</smil>
```

For More Information: For information on SMIL, see Chapter 7.

Secure Transactions

With Flash forms, you can build transaction functionality directly into Flash clips streamed to RealONE Player. This lets you add e-commerce capability to your presentation, for example. If the Flash clip connects to a secure server, RealONE Player transmits the encrypted information through the viewer's browser. RealONE Player does not support encryption directly, so the viewer needs to have a browser that supports it. Any encrypted response sent back by the secure server displays in the browser.

> **Note:** Because RealONE Player does not support secure transactions, you should not send an HTTP **POST** or **GET** command to a secure server if you intend for the server's response to come back to RealONE Player. For example, do not connect to a secure server by using Flash's **Load Variables** or **Load Movie** command.

Mouse Events

RealONE Player tracks certain mouse events differently than does the Flash plug-in used with browsers. Although this does not change how you build a streaming Flash presentation and it will not affect most viewers, you should be aware of this behavior.

The Flash browser plug-in records mouse events that occur outside of the Flash area. For example, a user may click and hold on an icon, drag the pointer out of the Flash area, and release the mouse button. In this case, the Flash browser plug-in knows that the mouse button has been released. RealONE Player, however, does not record mouse events that occur outside of its Flash region. Instead, it assumes that the button is still held down when the pointer returns to the Flash region.

Streaming a Flash Clip

This section summarizes the process for streaming a Flash clip. The Flash 5 program can export a RealAudio clip, a tuned Flash Player clip, and a SMIL file automatically. If you use Flash 5, refer to your Flash user's guide for instructions on exporting and tuning clips. Macromedia's Web site also provides information about exporting clips with Flash 5:

http://www.macromedia.com/support/flash/publishexport/realplayer/realplayer.html

If you are using a version of the Flash program prior to version 5, you must perform the following export and tuning steps manually.

➤ To create a streaming Flash clip manually:

1. Export the Flash Player clip.

   RealServer streams only the Flash Player format (.swf), which is a compressed version of the animation. You cannot stream the Flash source file format (.fla). If your animation includes a soundtrack, disable the audio stream when you export the clip. Refer to the Flash user manual for step-by-step instructions on the exporting a Flash Player clip.

   > **Note:** Keep in mind that RealONE Player plays the Flash 4, 3, and 2 Player formats. It does not play clips in the Flash 5 format.

2. Tune the Flash Player clip.

   With the Flash tuner, set the clip's streaming bit rate. This necessary step also eliminates bandwidth spikes that can cause rebuffering. The Flash tuner is included in the RealSystem Authoring Kit, available at this Web site:

   http://www.realnetworks.com/products/authkit/index.html

3. Export the soundtrack.

   If your animation includes a soundtrack, export the soundtrack as a Windows WAV file or Macintosh QuickTime file. If exporting to QuickTime (or any other video format), set low height and width attributes to minimize disk space use.

4. Encode the soundtrack as RealAudio.

   Encode the exported WAV or QuickTime soundtrack in the RealAudio format with RealProducer, using the file extension .rm.

5. Deliver the Flash presentation.

   Transfer your clips to RealServer. Then write the SMIL and Ram files necessary to stream the presentation.

   - Streaming a single Flash clip

     If your have a single Flash clip, your RealServer administrator can give you the URL to use in your Web page's hyperlink to the clip. If the RealServer does not use Ramgen, or you are delivering the clip through a Web server, you need to write a Ram file.

     For More Information: For more on RealServer and Ramgen, see "Using Ramgen for Clips on RealServer" on page 425. Ram files are described in "Launching RealONE Player with a Ram File" on page 420.

   - Streaming a Flash clip with another clip

     If your presentation has multiple clips, you write a SMIL file that organizes the presentation and gives the clip URLs. You next link your Web page to the SMIL file. In its simplest form, the SMIL file gives the full URLs to the clips and specifies that the clips play in parallel. The following example is for a Flash clip that plays in parallel with a RealAudio soundtrack:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
 <body>
  <par>
   <audio src="rtsp://realserver.example.com:554/media/sound.rm"/>
   <animation src="rtsp://realserver.example.com:554/media/cartoon.swf"/>
  </par>
 </body>
</smil>
```

     You can also use SMIL to define hypertext links, create timing offsets between clips, or add presentation information such as title, author, copyright, and abstract. For complete information on SMIL, see Chapter 7.

CHAPTER

6

REALTEXT MARKUP

With RealText, you can create timed text presentations that can stream alone or in combination with other media such as audio or video. This makes RealText a handy means for adding text to SMIL presentations. Because RealText creates timed text, you can use it to add subtitles to a video, for example, or to provide closed-captioning.

For More Information: Appendix E beginning on page 479 provides a quick reference to RealText tags and attributes.

Understanding RealText

Using any text editor, you can create a RealText clip in a text file that uses the file extension .rt. The file includes the text you want to display, as well as the RealText markup that describes how to display and time the text. Like a RealVideo or Flash clip, a RealText clip has a height and width, as well as an intrinsic duration, from a few seconds to several hours. The following are some of the features that RealText provides:

- Font, size, and color control

  The RealText markup lets you create text in many different fonts, sizes, and colors.

- Timing control

  RealText timing commands control when each paragraph, sentence, word, or letter appears. You might display a new sentence every few seconds, as in a video subtitle. Or you could make letters appear one at a time as if they were being typed across the screen.

- Flowing text

  Within a RealText clip, words can scroll up the screen or from side to side. This lets you create a window of smoothly flowing text. You can even make text loop, creating an endlessly flowing marquee.

- Positioning commands

  With the optional positioning commands, you can control exactly where each word appears within the RealText window.

RealText Language Support

RealText supports a number of languages, including English, Chinese, Korean, Japanese (Kanji), and many European languages. It can stream text in any language that can be written in one of its supported character sets, which are listed in the section "Specifying the Character Set" on page 106. Each character set supports at least one font, as described in "Setting the Font" on page 109.

> **Note:** Character set and font support is built into RealText. Therefore, RealText does not necessarily support all character sets and fonts supported by various Web browsers.

Structure of a RealText Clip

A RealText clip is a text file that uses the file extension .rt. At the top of the file you write a `<window>` tag that can include several attributes that set overall parameters, such as the window type, width, height, and duration. The file ends with a `</window>` tag. Between these tags, you add the text that you want to display in RealONE Player, using RealText tags and attributes to lay out and time the text. The following example is a simple RealText file that displays a new line of text every three seconds:

```
<window height="250" width="300" duration="15" bgcolor="yellow">
Mary had a little lamb,
<br/><time begin="3"/>little lamb,
<br/><time begin="6"/>little lamb,
<br/><time begin="9"/>Mary had a little lamb
<br/><time begin="12"/>whose fleece was white as snow.
</window>
```

Rules for RealText Markup

The RealText markup is similar to SMIL, and follows the same basic rules described in "Creating a SMIL File" on page 133. The following are the main points in mind when writing a RealText file:

- Use lowercase characters for RealText tags and attributes.
- A tag that does not have a corresponding end tag (for example, the `<ul>` tag has the end tag `</ul>`), closes with a forward slash, as in a `<br/>` tag, for example.
- Attribute values must be enclosed in double quotation marks.
- Save your RealText file with the file extension .rt. Do not include spaces in the file name. For example, you can have the file `my_realtext.rt` but not the file `my realtext.rt`.
- Use codes to include angle brackets, ampersands, or nonbreaking spaces as RealText display characters. See "Using Coded Characters" on page 118.
- As in HTML, you can add a comment to a RealText file like the following. Note that the comment tag does not need to close with a slash.

```
<!-- This is a comment -->
```

RealText Bandwidth

Because a RealText clip is a simple text file, it consumes minimal bandwidth and streams quickly to RealONE Player. RealText presentations are therefore easily accessible to viewers with slow network connections. When combining RealText with other clips, you need to ensure that RealText has approximately 1 Kbps of available bandwidth.

> **For More Information:** For more on bandwidth allocation, see "Step 3: Develop a Bandwidth Strategy" on page 31.

RealText in a SMIL Presentation

You can easily combine RealText with any other clip through a SMIL file. Chapter 7 explains the basics of SMIL. The section "Playing Clips in Parallel" on page 177 explains how to display RealText along with other clips. You'll also need to understand SMIL layouts as described in Chapter 11.

> **Tip:** To see examples of RealText displayed with other clips, get the HTML+Javascript version of this guide as described in

"How to Download This Guide to Your Computer" on page 5, and view the **Sample Files** page.

RealText Broadcast Application

RealText does not have to be created in a static file. A broadcast application can capture live text, add RealText markup to it, and send it to RealServer. A sample broadcast application is included with the RealSystem Authoring Kit, which is available for download at this Web address:

- **http://www.realnetworks.com/products/authkit/index.html**

You can also build a broadcast application with the RealSystem Software Development Kit (SDK) available for download at this Web page:

- **http://www.realnetworks.com/devzone/downlds/index.html**

Setting RealText Window Attributes

The <window> and </window> tags that begin and end a RealText file, respectively, set presentation attributes such as the window's height and width Here is an example of a <window> tag:

```
<window type="marquee" duration="2:05:00.0" underline_hyperlinks="false">
...all text and RealText markup...
</window>
```

You specify attributes in the form *attribute="value"* within the <window> tag, much as you specify HTML table attributes within the HTML <TABLE> tag. No attributes are required for the <window> tag, however. If you do not specify an attribute, the attribute's default value applies.

Specifying the Window Type

The <window> tag's type="*window type*" attribute sets specific properties for the RealText clip:

```
<window type="scrollingnews" ...>
```

Choose a window type depending on how you want to display text. Each window type has preset default values that make it easier to create certain types of text displays. You can create any type of RealText clip using just the default window type of generic, however. The following are the RealText window types:

- generic

  This is the default window type. You can use the generic window type to create any type of RealText clip based on the other attributes you include in the <window> tag.

- scrollingnews

  A scrollingnews window scrolls text upward at a specified rate for the entire presentation. The text initially appears at the top of the window.

- teleprompter

  A teleprompter window fills the display area with text starting at the top of the screen. As more timed text displays, the new text appears at the bottom of the screen and pushes older text up. The text does not scroll smoothly as in a scrollingnews window, though.

- marquee

  In a marquee window, text crawls from right to left and can loop. Text is centered vertically within the window.

- tickertape

  A tickertape windiw is like a marquee window, but text displays at the window's top or bottom edge, rather than in the center.

Window Type Default Values

Each window type sets a number of default values for the RealText clip. The following table lists the attribute default values that differ based on the choice of window type. Keep in mind that you can change any default value for any window type through the <window> tag. If you want a marquee window to be 320 pixels wide instead of 500 pixels, for example, you add width="320" to the <window> tag to override the window type's default width value.

Default Values for RealText Window Types

Value	generic	scrollingnews	teleprompter	marquee	tickertape
width in pixels (page 96)	320	320	320	500	500
height in pixels (page 96)	180	180	180	30	30
background (page 96)	white	white	white	white	black
horizontal crawl rate in pixels per second (page 100)	0	0	0	20	20

(Table Page 1 of 2)

Default Values for RealText Window Types (continued)

Value	generic	scrollingnews	teleprompter	marquee	tickertape
vertical scroll rate in pixels per second (page 100)	0	10	0	0	0
text looping (page 101)	no	no	no	yes	yes

(Table Page 2 of 2)

Setting the Window Size and Color

The `width` and `height` attributes determine the RealText window's width and height in pixels, respectively. The `bgcolor` attribute determines the window's background color. "Specifying RealText Color Values" on page 113 explains RealText color values. Here is an example that sets a window size and color:

```
<window width="400" height="225" bgcolor="blue"...>
```

> **For More Information:** Default values for size and background color are listed in the table "Default Values for RealText Window Types" on page 95.

Creating a Transparent Window Background

Using the `rn:backgroundOpacity` attribute in SMIL, you can turn the RealText window's background color fully transparent or semi-transparent. Within the RealText file, you define an opaque color, such as black or white, as the value of the `<window>` tag's `bgcolor` attribute. In your SMIL file, you then specify a percentage value for `rn:backgroundOpacity`. For more information on this SMIL attribute, see "Adjusting Transparency in a Clip's Background Color" on page 158.

> **Tip:** A semi-transparent background is useful when overlaying a video with RealText subtitles. To see an example of this, get the HTML+Javascript version of this guide as described in "How to Download This Guide to Your Computer" on page 5, and view the **Sample Files** page.

RealText Window Size and SMIL Region Size

When you add RealText to a SMIL presentation, you display your RealText clip in a SMIL region. For best results, create a SMIL region that is the same height and width as the RealText clip. Displaying a RealText clip in a SMIL region that is larger or smaller than the clip may enlarge or shrink the text,

depending on how you set the <region> tag's fit attribute. The sections below explain which fit values are best to use.

Note that enlarging or shrinking a RealText clip through SMIL does not affect line breaks. Line breaks are determined by the RealText window's width, font, and font size. You could place a RealText window that is 200 pixels wide in a SMIL region that is 150 pixels wide, for example, and scale the clip's width down by adding fit= "fill" to the SMIL <region> tag. This simply makes all the text smaller. It does not cause lines to break at different places within the text.

> **For More Information:** SMIL regions are described in "Playback Regions" on page 196. The section "Defining How Clips Fit Regions" on page 227 explains the <region> tag's fit attribute. RealText word wrapping is described in "Wrapping Text to New Lines" on page 101.

When a SMIL Region is Larger than the RealText Clip

When the SMIL region is larger than the RealText clip, the default value fit="hidden" is recommended for the <region> tag. This keeps the RealText clip as its specified size. You can then use a registration point, as described in "Creating Registration Points" on page 221, to position the clip within the region. The registration point might center the clip in the region, for example.

If you want to scale the RealText clip larger, using fit="meet" in the <region> tag typically gives the best results because it preserves the clip's aspect ratio. This scales the text larger but maintains the relative letter spacing. You can use fit="fill" to make the RealText clip the same size as the region, but distortion in letter spacing may make the clip unreadable if the region has a markedly different height-to-width ratio than the clip.

When a SMIL Region is Smaller than the RealText Clip

When the SMIL region is smaller than the RealText clip, the default value fit="hidden" in the <region> tag may prevent some text from displaying. The value fit="meet" is generally the best choice, because it scales the clip smaller to fit completely inside the region while preserving the relative letter spacing. When displaying RealText in a smaller region, though, you need to be careful to keep the text from scaling down to an unreadable size.

Setting the Clip Duration

The duration attribute specifies how long the RealText clip plays. The default is 60 seconds. RealText uses only the "normal play time" timing values of

hh:mm:ss.xy, which are described in "Using the Normal Play Time Format" on page 240. In this timing method, only the ss field is required. For example, the following duration attributes make the clip last 90 seconds:

```
<window duration="90" ...>
```

5 and 1/2 minutes:

```
<window duration="5:30" ...>
```

and 1 hour, 33 minutes, and 15 seconds:

```
<window duration="1:33:15" ...>
```

RealText Durations and SMIL Durations

When you put RealText in a SMIL presentation, SMIL timing values can override the duration defined in the RealText clip. Suppose a RealText clip named marquee.rt has a duration of three minutes:

```
<window duration="3:00.0" ...>
```

If you put this clip into a SMIL presentation with the following SMIL clip source tag, the dur="2min"attribute tells RealONE Player to stop playing this clip after two minutes regardless of the clip's internal timeline:

```
<textstream src="rtsp://realserver.example.com/marquee.rt dur="2min" .../>
```

If the SMIL duration is longer than the RealText duration, a fill attribute can specify how RealONE Player treats the clip once it has stopped playing:

```
<textstream src="rtsp://realserver.example.com/marquee.rt dur="4min" fill="freeze"/>
```

> **For More Information:** For more on the SMIL fill attribute, see "Setting a Fill" on page 251.

Tips for Setting RealText Clip Durations

- When you work with both SMIL and RealText, be careful not to confuse the different duration attributes. In RealText, the duration attribute must be duration, whereas in SMIL it must be dur.
- RealText uses only the normal play time format (hh:mm:ss.xy) for setting time values. It cannot use SMIL timing shorthand values such as "3min".
- If your RealText clip stops before all text has displayed, the duration time is probably set too low. To help prevent this problem, set a high duration when you start writing your RealText markup. Then set the final duration time when you have finished defining the RealText markup.

- The final duration should be slightly higher than the time it takes to display all the text. If all text displays within two minutes, for example, set a duration of two minutes and five seconds.
- Setting a duration much higher than the time it takes all text to display may unnecessarily delay clips that play after the RealText clip in a SMIL sequence, and can make it difficult for viewers to use the RealONE Player position slider to search for specific parts of the RealText clip.
- The duration time you set is reflected in RealONE Player. If you set a duration of five minutes, for instance, the RealONE Player status bar lists the clip length as `5:00.0` and the RealONE Player position slider takes five minutes to travel from left to right.
- Text is not erased at the end of a RealText clip's duration. The final text remains in the RealText window unless you erase the text with a `<clear/>` tag, or the text moves out of the window because you have set a `scrollrate` or a `crawlrate`.

For More Information: See "Clearing Text from the Window" on page 104 and "Setting a Scroll Rate or a Crawl Rate" on page 100.

Adding a Version Number

The `<window>` tag can include a version number, as shown in this example:

```
<window version="1.5"...>
```

You typically do not have to specify a version number when using RealText in English. Properly displaying languages other than English may require that you specify a version number explicitly in the `<window>` tag, however. This chapter tells you when a version number is required to use a specific feature. The following paragraphs summarize the features that require you to add version numbers:

- `version="1.2"`

  This RealText version provides support for the `mac-roman` character set, and changes the default character set from `us-ascii` to `iso-8859-1`. This version requires RealPlayer 7 or later, so RealPlayer G2 will autoupdate to the latest version of RealONE Player before playing the clip.

- version="1.4"

  This RealText version provides support for the iso-2022-kr character set and the Korean language. This version requires RealPlayer 7 or later, so RealPlayer G2 will autoupdate to the latest version of RealONE Player before playing the clip.

- version="1.5"

  This RealText version supports hyperlinks in the format *protocol:path*, as explained in "Issuing RealONE Player Commands" on page 117. This version requires RealPlayer 8 or later, so RealPlayer G2 and RealPlayer 7 will autoupdate to the latest version of RealONE Player before playing the clip.

> **Tip:** Because newer versions of RealText encompass all features from previous versions, you can always specify a higher version than that required for a feature. If a feature requires RealText version 1.2, for example, you can use 1.5 as the version number.

Specifying Hyperlink Appearance

The underline_hyperlinks="true|false" attribute determines whether hyperlinks are underlined. The default is true. The link="*color*" attribute, which defaults to blue, sets the color of hyperlinks within the text. Here is an example:

```
<window underline_hyperlinks="false" link="red" ...>
```

> **For More Information:** See "Specifying RealText Color Values" on page 113 for color options.

Controlling Text Flow

As described in the following sections, several <window> tag attributes (scrollrate, crawlrate, wordwrap, loop, and extraspaces) affect how text displays in the RealText clip.

Setting a Scroll Rate or a Crawl Rate

The scrollrate attribute sets the number of pixels per second that the text scrolls from the bottom of the window to the top for the duration of the clip. It has no effect on tickertape and marquee windows. Here is an example:

```
<window scrollrate="25" ...>
```

The crawlrate attribute specifies the number of pixels per second that the text moves horizontally from right to left for the duration of the clip. Here is an example:

```
<window crawlrate="40" ...>
```

> **Tip:** A RealText clip should not use both scrollrate and crawlrate. For best results, use a scrollrate or a crawlrate under 30. The best values are 25, 20, 10, 8, 5, 4, 2, and 1. For rates faster than 30, use multiples of 20 or 25, such as 40, 50, 60, 75, 80, and so on.

> **For More Information:** The table "Default Values for RealText Window Types" on page 95 lists the default values for scrollrate and crawlrate in the standard window types.

Wrapping Text to New Lines

The wordwrap="true|false" attribute, which defaults to true, specifies whether word wrap is performed. When word wrap is on, text lines longer than the specified window width wrap to the following line. If it is off, long lines are truncated by the window border. This attribute has no effect for windows that have horizontal text motion, such as a marquee window.

Looping Text

The loop="true|false" attribute is available only in tickertape and marquee windows, which have horizontal "crawling" motion. In these window types, the loop attribute defaults to true, which tells RealONE Player to redisplay ("loop") text under these circumstances:

- In a clip that does not use <time begin="..."> tags to set begin times on text blocks, looping occurs if all text has moved out of the window but the clip's duration has not expired. If the duration is two minutes but all text has moved out the window after one minute, for example, the text begins again.
- In a clip that uses <time begin="..."> tags to set begin times, text blocks loop if they have scrolled out of the window and the next text block's begin time has not elapsed. For example, consider this markup:

  ...*first text block*...<time begin="1:00.0"><clear/>
...*second text block*...

In this case, the first text block loops as necessary for one minute. At that time, the <clear/> tag erases the window and the <br/> tag starts the second text block at the window's right-hand side.

> **For More Information:** For information on timing and erasing text, see "Timing and Positioning Text" on page 102. The <br/> tag is described in "Adding Space Between Text Blocks" on page 113.

- In a RealText broadcast, text loops as necessary until new text arrives. If the text is looping as the new text arrives, the new text displays as soon as the old text has moved out of the window. The new text then becomes part of the loop.

Ignoring Extra Spaces

When set to its default value of use, the extraspaces="use|ignore" attribute makes RealText recognize all blank spaces between text chunks and markup tags. If three spaces occur between two words in the RealText file, for example, RealONE Player displays all three spaces. It treats each carriage return and tab as a space.

If you specify extraspaces="ignore", RealONE Player treats spaces, tabs, line feeds, and carriage returns as does a Web browser, except when they are between the <pre>...</pre> tags. When spaces or carriage returns occur contiguously in the text, RealONE Player interprets them as a single space, no matter how many of them are present. So in this case, three contiguous spaces display as one space in RealONE Player.

> **For More Information:** The <pre>...</pre> tags are described in the section "Preformatting Text" on page 114

Timing and Positioning Text

The following sections explain the RealText tags you can use between the <window> and </window> tags to control when and where text appears within the RealText window.

Controlling When Text Appears and Disappears

The <time/> tag controls the RealText presentation timeline by specifying when text blocks appear or disappear. The <time/> tag is useful primarily in

RealText clips in which text does not scroll or crawl across the screen. In these clips, RealONE Player displays all text as quickly as it can if you do not time the text with <time/> tag.

The <time/> tag can have two attributes, begin and end. You can use one or both attributes in each <time/> tag. Each attribute specifies a time when the text appears or disappears, respectively. As with the <window> tag's duration attribute, a <time/> tag specifies a time in the "normal play time" format:

```
<time begin="hh:mm:ss.xy"/>
<time end="hh:mm:ss.xy"/>
```

In the following sample text block, the first phrase appears at the start of the RealText presentation. The subsequent text blocks appear at three seconds into the timeline, and six seconds into the timeline, respectively:

```
Mary had a little lamb, <time begin="3"/>little lamb, <time begin="6"/>little lamb.
```

For More Information: See "Using the Normal Play Time Format" on page 240 for more on <begin> tag timing values.

Using an End Time

Text with an end time is erased when the specified end value is reached. Otherwise it stays active until the presentation ends or the entire window is erased with <clear/>. In the following example, text blocks begin at different times, but all end at the same time. Note that just as with a begin time, an end time must appear before the text block in the file:

```
<time end="25"/><time begin="5"/>This text starts to display at 5 seconds.
<time begin="10"/>A new line appears each additional 5 seconds.
<time begin="15"/>But all this text disappears ...
<time begin="20"/>at 25 seconds into the clip.
```

You can also combine the begin and end attributes in a single <time/> tag as shown here:

```
<time begin="23" end="55.5"/>This text displays 23 seconds into the presentation
and disappears at 55.5 seconds.
```

It's important to note that text following a <time/> tag has the specified begin or end value until a new value is given. Therefore, once you specify an end time for a text block, you must specify an end time for all following blocks. For example, the following text would not display properly:

```
<time begin="23" end="55.5"/>Display at 23 seconds in.
<time begin="56"/>Display at 56 seconds in.
```

Because the second line in the preceding example does not include an end time, the previous end time of 55.5 still applies. The second line cannot be displayed, however, because its begin time is later than its end time.

Tips for Using <time/> Tags

- The <time/> tags are not necessary in a window with a scrollrate or crawlrate unless you want to delay text, have it become visible after it has moved into the window, or have it disappear before it moves out of the window. See also "Looping Text" on page 101 for information on how <time/> tags can affect text looping.
- To freeze text on the screen after the clip's duration has elapsed, do not set an end time. Or, have the end time exceed the window's duration as shown in this example:

```
<window duration="30" ...>
  ...some text elements...
  <time begin="25" end="31"/>Text that stays frozen onscreen.
</window>
```

- To replace a line of text with a new line every few seconds (as in video subtitles), do not use end times. For each new line of text, set the appropriate begin time followed by a <clear/> tag, as described below.

Clearing Text from the Window

The <clear/> tag removes all text from the window. The text that follows this tag then displays at the window's normal starting point, which is typically the window's top or right edge, unless you position the text elsewhere. You can add <clear/> after <time begin="..."/> to erase text before displaying new text. This is often an easier method of removing text than using <time end="..."/> tags. In the following example, each new line erases the preceding line:

```
<time begin="5"/>This line displays at 5 seconds.
<time begin="10"/><clear/>This line erases the previous line at 10 seconds.
<time begin="15"/><clear/>This line erases the previous line at 15 seconds.
<time begin="20"/><clear/>This line erases the previous line at 20 seconds.
```

A <clear/> tag removes all preceding text, even text that has an end time that has not yet elapsed. In the following example, the second line of text is set to end at 20 seconds. However, the <clear/> tag appears at 15 seconds into the presentation and clears this line, eliminating the end time for all following text:

```
<time begin="5"/>They all lived happily.
<time begin="10" time end="20"/>And so our story ends.
<time begin="15"/><clear/>Goodbye!
```

> **Note:** The <clear/> tag does not reset text appearance. For example, if text appears bolded before the <clear/> tag, it remains bolded after the <clear/> tag.

Positioning Text in a Window

These <pos/> tag can position text anywhere in the RealText window. You can use its x attribute for horizontal positioning, and its y attribute for vertical positioning. Each attribute takes a value in pixels, as shown in these examples:

```
<pos x="10"/>
<pos y="25"/>
```

A <pos y="*pixels*"/> tag moves the upper, left corner of the subsequent text block the specified number of pixels down from the window's top edge. A <pos x="*pixels*"/> tag indents the text block the specified number of pixels in addition to the two-pixel default padding that applies to all text blocks. You can combine both tags in a single tag like this:

```
<pos x="10" y="25"/>
```

> **Note:** These tags work only if scrollrate and crawlrate are both 0 (zero). For more on these attributes, see "Setting a Scroll Rate or a Crawl Rate" on page 100.

Aligning Text in a Tickertape Window

Th <tu>...</tu> and <tl>...</tl> tag sets function only with tickertape windows. They display the enclosed text at the window's upper (<tu>...</tu>) or lower (<tu>...</tu>) edge. Optionally, they can include a color attribute that specifies the color for the text, as shown in this example:

```
<tu color="blue">...text to display at tickertape window's upper edge...</tu>
<tl color="yellow">...text to display at tickertape window's lower edge...</tl>
```

When a tag specifies a color with the color attribute, the color applies to text enclosed by all subsequent tags of that type until another tag of that type changes the color. However, color specified for <tu> elements does not affect color for <tl> elements, and vice versa. The default color for <tu> elements is white, the default for <tl> elements is green.

For More Information: Refer to "Specifying RealText Color Values" on page 113 for information about choosing colors.

Ensuring Text Delivery

Use the <required> and </required> tags to enclose text that must be delivered to RealONE Player under any circumstance. During extremely adverse network conditions, RealSystem will halt the presentation if necessary rather than drop the text. You can use these tags sparingly, though, because RealSystem normally ensures that very little data loss occurs in transmission.

Note: Although RealServer provides reliable streaming, packets not marked as required may be lost occasionally. If a block of text does not get through, RealONE Player displays a red ellipsis (**...**) to indicate missing text.

Specifying Languages, Fonts, and Text Colors

The RealText <font> tag controls the text font and color. Because it also specifies the character set, it determines which languages you can write in. As shown in the following example, the <font> tag can take multiple attributes, and it always uses an end tag:

```
<font size="+2" face="Courier New" color="red">...text...</font>
```

Multiple <font> tags can also be nested to turn various font features on and off:

```
<font attribute="A">...turn on font attribute "A"...
<font attribute="B">...turn on font attribute "B"...
</font>...turn off font attribute "B"...
</font >...turn off font attribute "A"...
```

Specifying the Character Set

With the <font> tag's charset attribute, you can control the character set used to display the text. You do not need to specify the character set explicitly to write text in English. However, you may need to specify the character set to write text in other supported languages. You can set the character set as well as the font face immediately after the <window> tag within a RealText file, as shown in the following example:

```
<window version="1.4"...>
<font charset="iso-2022-kr" face="Batang">
...Korean text that uses the iso-2022-kr character set and Batang font...
</font>
</window>
```

You can also use multiple <font> tags to change character sets within a RealText file and display text in different languages:

```
<font charset="iso-2022-kr" face="Batang">
...Korean text that uses the iso-2022-kr character set and Batang font...
</font>
<font charset="x-sjis" face="Osaka">
...Kanji text that uses the x-sjis character set and Osaka font...
</font>
```

It is important to note that RealText always uses its specified character set, not the default character set of the computer playing the clip. In RealText version 1.2 and higher, the default character set is iso-8859-1. To display Korean text on a machine that uses the iso-2022-kr character set by default, for instance, you must explicitly set charset="iso-2022-kr" in the RealText <window> tag. If you do not, RealText will use its default iso-8859-1 character set, even though iso-2022-kr is the machine's default.

> **Note:** If the computer does not recognize the character set specified in the RealText clip, it displays the text in its default character set. The result is typically unreadable.

> **For More Information:** As noted in the following sections, using some character sets requires you to include a version number in the <window> tag. For more on version numbers, see "Adding a Version Number" on page 99.

us-ascii

The us-ascii character set is the default character set used with most RealText fonts when no version number is specified in the <window> tag.

iso-8859-1

The iso-8859-1 character set is identical to us-ascii, but includes support for accented characters (upper 128 characters) used in many European languages. This is the default character set used when you specify version="1.2" or higher in the <window> tag. Use it when writing accented European languages on a

Windows or Unix computer. The following languages can be represented with the iso-8859-1 character set:

Afrikaans	Basque	Catalan	Danish	Dutch	English
Faeroese	Finnish	French	Galician	German	Icelandic
Irish	Italian	Norwegian	Portuguese	Spanish	Swedish

> **Note:** The ISO-8859 standard specifies several additional character sets, such as iso-8859-2 and iso-8859-3. RealText supports only iso-8859-1, however, meaning that Cyrillic, Arabic, Greek, Hebrew, and several Eastern European languages are not supported in RealText.

mac-roman

Use the mac-roman character set when writing in an accented European language on a Macintosh computer. Using this character set ensures that marks such as umlauts (for example, "ü") display properly when the RealText clip plays on a Windows or Unix computer. Use version="1.2" or higher in the <window> tag to handle this character set correctly.

> **Note:** You do not need to use the mac-roman character set when writing in English. When writing in accented languages on a Windows or Unix machine, use the iso-8859-1 character set instead.

x-sjis

The x-sjis character set is for Kanji and the Osaka font. Use version="1.2" or higher in the <window> tag to handle this character set correctly.

gb2312

The gb2312 character set is for Simplified Chinese.

big5

The big5 character set is for Traditional Chinese.

iso-2022-kr

The iso-2022-kr character set is for Korean. Use version="1.4" or higher in the <window> tag to handle Korean text correctly.

Setting the Font

This tag attribute face="*font name*" controls the font use. You can use any number of fonts in the same RealText clip. When switching fonts, be sure to turn off the preceding font with a tag, as shown in this example:

```
<font face="Arial">
...Text in the Arial font...
</font>
<font face="Garamond">
...Text in the Garamond font...
</font>
```

Font faces correspond to character sets as described in the section "Specifying the Character Set" on page 106. For non-Western fonts, you must specify the correct character set for the font to display properly. If you specify no font, RealText uses the Times New Roman or Times font regardless of the character set specified.

English and European Language Fonts

When writing in English or European languages, use a font name from the "Windows Font Name" column of the following table, which lists fonts that use the us-ascii or iso-8859-1 character set. If the specified font isn't available on a Macintosh or Unix computer, RealText uses a system font as indicated in the table below. For example, a RealONE Player on a Macintosh displays text in Courier if the Algerian font is not available. The notation "(always)" indicates cases where RealText always defaults to a system font. For example, the Fixedsys font always displays as Courier on a Macintosh.

RealText Font Support for us-ascii and iso-8859-1 Character Sets

Windows Font Name	Macintosh Default if Font not Available	Unix Default if Font not Available
Algerian	Courier	Courier
Arial	Helvetica	Helvetica
Arial Black	Helvetica	Helvetica
Arial Narrow	Helvetica	Helvetica
Arial Rounded Mt Bold	Helvetica	Helvetica
Book Antiqua	Helvetica	Helvetica
Bookman Old Style	Helvetica	Helvetica
Braggadocio	Helvetica	Helvetica

(Table Page 1 of 2)

RealText Font Support for us-ascii and iso-8859-1 Character Sets

Windows Font Name	Macintosh Default if Font not Available	Unix Default if Font not Available
Britannic Bold	Helvetica	Helvetica
Brush Script	Times	Times
Century Gothic	Helvetica	Helvetica
Century Schoolbook	Helvetica	Helvetica
Colonna Mt	Times	Times
Comic Sans Ms	Times	Times
Courier New	Courier	Courier
Desdemona	Helvetica	Helvetica
Fixedsys	Courier (always)	Courier
Footlight Mt Light	Helvetica	Helvetica
Garamond	Times	Times
Haettenschweiler	Helvetica	Helvetica
Helvetica (Arial is used if Helvetica is not found.)	Helvetica	Helvetica
Impact	Helvetica	Helvetica
Kino Mt	Times	Times
Matura Mt Script Capitals	Times	Times
Modern	Helvetica	Helvetica
Ms Dialog	Times	Times
Ms Dialog Light	Times	Times
Ms Linedraw	Helvetica	Helvetica
Ms Sans Serif	Helvetica	Helvetica
Ms Serif	Helvetica	Helvetica
Ms Systemex	Times	Times
Playbill	Times	Times
Small Fonts	Times	Times
System	Geneva (always)	Times
Terminal	Geneva	Times
Times New Roman	Times (always)	Times
Verdana	Helvetica	Helvetica
Wide Latin	Helvetica	Helvetica

(Table Page 2 of 2)

Tip: A Macintosh that has Microsoft Internet Explorer 4.0 or later installed should have most of the Windows fonts.

Asian Language Fonts

RealText also supports the following fonts that use character sets other than us-ascii and iso-8859-1.

RealText Font Support for Non-Western Character Sets

Font Name	Characters	RealText Font Face Tag	charset
AppleGothic	Korean		iso-2022-kr
Batang	Korean		iso-2022-kr
BatangChe	Korean		iso-2022-kr
Gothic	Korean		iso-2022-kr
Gulim	Korean		iso-2022-kr
GulimChe	Korean		iso-2022-kr
Osaka	Kanji		x-sjis
Seoul	Korean		iso-2022-kr
’宋体	Simplified Chinese	 (The face name displays as gibberish without the gb2312 character set.)	gb2312
細明體	Traditional Chinese	 (The face name displays as gibberish without the big5 character set.)	big5

Note: Korean and Japanese are supported in RealONE Player for Windows and Macintosh, but not for Unix.

Setting the Text Size

The tag attribute size="*n*" lets you control the font size, as shown in this example:

...*text that is one size larger*...

You can use relative sizes or absolute sizes as shown in the table below. This table also lists the height in pixels for each size. The pixel sizes are for reference only. You cannot specify a pixel size directly in RealText.

Font Sizes

Relative Size	Absolute Size	Pixel Size Reference
-2	1	12 pixels
-1	2	14 pixels
+0 (default)	3	16 pixels
+1	4	20 pixels
+2	5	24 pixels
+3	6	36 pixels
+4	7	48 pixels

Note: You can also specify relative sizes smaller than -2 or larger than +4, but they are treated as -2 and +4, respectively.

Controlling Text Colors

Two attributes of the <font> tag, color and bgcolor, let you set the color for the text letters, and the background against which the text appears. The section "Setting the Window Size and Color" on page 96 explains how to set the RealText window's background color.

Setting Text Letter Colors

The color attribute of the <font> tag lets you control the text color. It has no effect on tickertape windows because the <tu> and <tl> tags, which are described in "Aligning Text in a Tickertape Window" on page 105, set the tickertape text colors. The following example shows the text color set to red:

```
<font color="red">...red text...</font>
```

Creating Text Background Colors

Use the bgcolor attribute to the <font> tag to set the text background color. The default background color for text is "transparent", making the text background the same color as the window. The following example sets the text background to yellow:

```
<font bgcolor="yellow">...text with yellow background...</font>
```

Note that the text background color is independent of the window background color. If the window background color is blue, for example, and the text background color is yellow, a stripe of yellow appears on top of the blue window wherever the affected text displays. Within that yellow stripe, the text appears in the color set by the color attribute.

Specifying RealText Color Values

For RealText window backgrounds and fonts, you can use red/green/blue hexadecimal values (#RRGGBB), as well as the following color names, listed here with their corresponding hexadecimal values:

white (#FFFFFF)	silver (#C0C0C0)	gray (#808080)	black (#000000)
yellow (#FFFF00)	fuchsia (#FF00FF)	red (#FF0000)	maroon (#800000)
lime (#00FF00)	olive (#808000)	green (#008000)	purple (#800080)
aqua (#00FFFF)	teal (#008080)	blue (#0000FF)	navy (#000080)

> **Tip:** Appendix D provides background on hexadecimal color values. Note, though, that RealText does not support RGB color values used with SMIL.

Using Transparency as a Color

For text backgrounds, you can use bgcolor="transparent". This is the default for text backgrounds, meaning that the words following the tag do not have a colored rectangle drawn behind them, so the window background color shows around the letters. This lets you draw text on top of previous text (using the <pos/> tags) without "erasing" the previous text.

Controlling Text Layout and Appearance

The following tags let you lay out text in the RealText clip. Many of these tags are similar to HTML tags, and are provided for compatibility. However, unlike in HTML, RealText tags are case sensitive and a closing tag is always required. You cannot use a tag without a tag, for example, or use capital letters as in <P> and </P>.

Adding Space Between Text Blocks

The following tags add space between text blocks. If text flows across the screen horizontally, however, line breaks are not created.

...

The ... tags turn the enclosed text into a pargraph. In tickertape and marquee windows, it causes the text that follows it to display at the window's right edge. In all other window types, the and each cause the next text block to display two lines down.

The <br/> tag adds space between text. In tickertape and marquee windows, it causes the text that follows it to display at the window's right edge. In all other window types, this tag causes the text that follows to display on the next line.

Centering Text

The ... tags center the enclosed text. These tags behave the same as HTML centering tags, but they have no effect in windows with horizontal motion, such as tickertape and marquee windows. The tag forces a line break if and only if a line break caused by a tag such as <br/>, , or <hr/> does not immediately precede it. The tag always causes a line break.

> **Note:** RealText does not center text until it has determined the line length. In rare instances, one streamed packet may contain the first part of the line while another packet received several seconds later contains the end of the line. In this case, the first part displays flush left, and the entire line is centered and redisplayed when the second packet arrives.

Preformatting Text

The <pre>...</pre> tags work the same as in HTML. Text tagged with <pre> uses the Courier font at the current size. To change the font size, precede the preformatted block with a <font size="*n*"> tag. Line breaks, spaces, and tabs are preserved, with tabs defaulting to 64 pixels for 16 point text (the normal point size). Tab spaces are determined by dividing the text height by 2, then multiplying by 8.

> **For More Information:** For information on text heights, see the table "Font Sizes" on page 112. See also "Ignoring Extra Spaces" on page 102.

Using HTML-Compatible Tags

The following RealText tags are provided for HTML compatibility, allowing you to convert HTML to RealText more easily, and vice versa. These tags do not function the same in RealText as they do in HTML, however.

...

The <ol>... tags are for compatibility with HTML lists. Text between these tags is indented, but not numbered.

...

The <ul>... tags are for compatibility with HTML lists. Text between these tags is indented, but not bulleted.

...

The <li>... tags are for compatibility with HTML lists. They act like a <br/> tag.

<hr/>

The <hr/> tag is for compatibility with HTML horizontal rules. It acts like two <br/> tags.

Emphasizing Text

The following RealText tags let you add emphasis to text.

...

The ... tags display the enclosed text **bolded**.

...

The ... tags display the enclosed text *italicized*.

<s>...</s>

The <s>... tags ~~strike-through~~ the enclosed text.

<u>...</u>

The <u>...</u> tags display the enclosed text <u>underlined</u>.

Creating Links and Issuing Commands

The following sections describe tags you can use to launch URLs in a browser or RealONE Player. You can also use tags to issue RealONE Player commands such as **Pause** and **Play**.

> **Tip:** Link text is the color specified in the link attribute of the <window> tag. The link is underlined unless the <window> tag includes underline_hyperlinks="false".
>
> **For More Information:** SMIL files can also define hypertext links that may override the link you set here. For more information, see Chapter 14.

Creating a Mail Link

This tag turns the enclosed text into an e-mail hyperlink:

```
<a href="mailto:address">...</a>
```

When the viewer clicks the link, RealText passes the e-mail address to the viewer's browser. Use an address in the standard form, such as name@company.com. If the browser is configured for e-mail, the e-mail client opens a new message with the defined address in the "to" line.

Opening a URL in RealONE Player or a Browser

The following RealText tag makes the text enclosed between <a href...> and a hyperlink that opens in a browser or RealONE Player:

```
<a href="URL" [target="_player"]>...</a>
```

The specified URL should begin with a protocol designation such as http:// or rtsp://. The optional target="_player" attribute launches the new stream in the current RealONE Player window. If you do not use the target attribute or you specify target="_browser", the linked URL opens in a Web browser window.

Example 1: Opening a URL in RealONE Player

The following example launches a new SMIL Presentation in RealONE Player, replacing the currently playing presentation:

```
<a href="rtsp://realserver.example.com/video2.smil target="_player">Play Next</a>
```

You can also open a link in a new RealONE Player media window. This lets you keep navigation information visible in one window, for example, while content

plays in another window. For information on doing this, see "Opening a New Media Window Through RealText or Flash" on page 306

Example 2: Opening a URL in a Web Browser

This example opens a URL in the user's browser:

```
<a href="http://realguide.real.com">Visit RealGuide</a>
```

For static files, you can also specify URLs relative to the location of the RealText source file. For example, the link `<a href="more.htm">...</a>` opens the file `more.htm` in the same directory as the RealText file. Relative links follow the standard HTML directory syntax.

Example 3: Opening a URL in the Form protocol:path

If you include `version="1.5"` (or higher if using a newer version of RealText) in the `<window>` tag, you can pass the browser a URL in the form *protocol:path* instead of *protocol://path*. Protocols using this format include those for Telnet and AOL Instant Messenger. For example, here is a RealText link that launches AOL Instant Messenger:

```
<window version="1.5"...>
...<a href="aim:goim?screenname=[name]">Send Me an Instant Message</a>...
</window>
```

Issuing RealONE Player Commands

The following tag makes the enclosed text a hyperlink that, when clicked, executes a RealONE Player command:

```
<a href="command" target="_player">...</a>
```

The commands are case-sensitive and must be enclosed in double quotes. The `target="_player"` attribute is required.

Seeking Into a Presentation

The following command instructs RealONE Player to seek to the specified time in the current text stream:

```
<a href="command:seek(time)" target="_player">Seek</a>
```

For example, the following instructs RealONE Player to seek to 1:35.4 in the stream:

```
<a href="command:seek(1:35.4)" target="_player">Seek</a>
```

Pausing a Presentation

When clicked, the following link causes RealONE Player to pause the stream:

```
<a href="command:pause()" target="_player">Pause</a>
```

Resuming Playback

Clicking the next link causes RealONE Player to begin or resume playing the stream:

```
<a href="command:play()" target="_player">Resume</a>
```

Using Coded Characters

The following table lists the character codes you can include in a RealText source file. Codes begin with an ampersand ("&") and end with a semicolon (";"). RealText interprets these characters the same way as popular Web browsers.

RealText Coded Character Set

Code	Displays as
`<`	<
`>`	>
`&`	&
` `	(nonbreaking space)
` ` to `ÿ`	Characters taken from the active character set as specified by the active `<font charset="...">` tag. The default character set is `iso-8859-1`, which is also known as ISO Latin 1. For a list of these characters, see the W3C reference at **http://www.w3.org/MarkUp/html-spec/html-spec_13.html**. See below, however, if you're using the `mac-roman` character set.

> **Tip:** The HTML+Javascript version of this manual includes a JavaScript file that generates the character codes for you. See "How to Download This Guide to Your Computer" on page 5 for details about getting the zipped HTML manual.

For example, the following RealText source text:

```
This is a bold tag: "&lt;b&gt;".
```

is displayed in a RealText window as:

```
This is a bold tag: "<b>".
```

Using Coded Characters with the mac-roman Character Set

Unlike HTML, RealText allows you to change character sets within a document. It then takes coded characters from the active character set. Generally, character codes 128 and lower are the same in all Western-language character sets. Those higher than 128 may differ, though. In the `mac-roman` character set, for example, `¦` is a paragraph symbol. But in `iso-8859-1`, this symbol is `¶`.

See **http://czyborra.com/charsets/mac-roman.gif** for a GIF chart of the `mac-roman` upper character set. Go by this chart, rather than the W3C reference provided above if you've set `<font charset="mac-roman">` and are entering coded characters of `` or higher. The values in the chart are in hexadecimal (base 16). The chart cell in the upper, left-hand corner equals 128 in decimal (base 10), so you can count across from there. To make a paragraph symbol when using `mac-roman`, for instance, you use `¦` in the RealText file because hexadecimal A6 on the chart is decimal 166.

RealText Examples

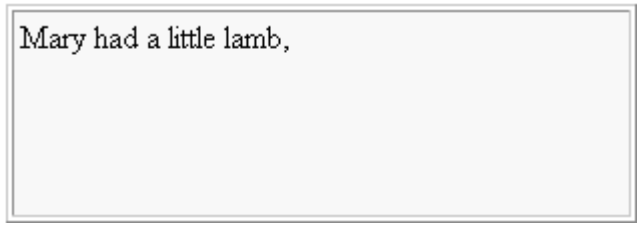
This following sections provide examples of how to create various types of RealText clips.

Generic Window

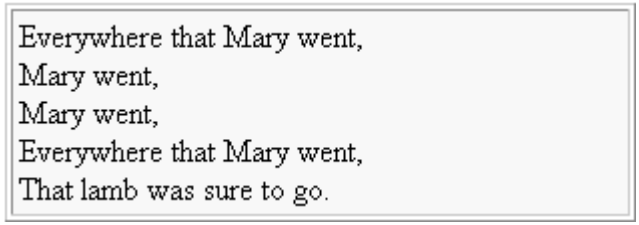
The following sample RealText markup creates a generic RealText window:

```
<window duration="30" bgcolor="yellow">
Mary had a little lamb,
<br/><time begin="3"/>little lamb,
<br/><time begin="6"/>little lamb.
<br/><time begin="9"/>Mary had a little lamb,
<br/><time begin="12"/>whose fleece was white as snow.
<br/><time begin="15"/><clear/>Everywhere that Mary went,
<br/><time begin="18"/>Mary went,
<br/><time begin="21"/>Mary went,
<br/><time begin="24"/>Everywhere that Mary went,
<br/><time begin="27"/>That lamb was sure to go.
</window>
```

When RealONE Player processes this file, it displays only the first line of the text from zero to three seconds into the stream:



Every three seconds after the first line displays, a new line appears as specified by <time begin="*n*"/>. At 15 seconds, <clear/> clears the displayed text and resets the text "cursor" to the upper-left corner of the window. When the stream finishes, all lines of text following the last <clear/> tag appear in the window:



Note the following about this sample clip:

- Because it was not specified in the <window> tag, word wrapping defaults to true. However, word wrapping is not necessary because <br/> tags force line breaks.
- <time/> tags need not appear after a <br/> tag. They can appear anywhere in the text.
- The example could have used <time end="…"/> tags to make individual lines of text disappear before the <clear/> tag cleared all the lines.

Tickertape Window

The following example shows the RealText markup for a tickertape window. This is the RealText (.rt) source file:

```
<window type="tickertape" duration="1:00" width="500" loop="true"
underline_hyperlinks="false" link="white">
<br/>
<b>
<tu><a href="http://www.dowjones.com/">DJIA</a></tu>
```

```
<tl>7168.35 +36.5 </tl>
<tu>NIKEI 225 Index</tu>
<tl>20603.71 +203.11</tl>
</b>
</window>
```

This source file produces the following window in RealONE Player:

Note the following about this sample clip:

- The text crawls from right to left at 20 pixels per second, the default `crawlrate` for a tickertape window.
- The `<b>` tag at the start **bolds** all following text.
- `<a href="http://www.dowjones.com/">DJIA</a>` makes `DJIA` a hyperlink that, when clicked, opens a browser to **http://www.dowjones.com/**.
- `DJIA` is not underlined because `underline_hyperlinks="false"` is declared in the `<window>` tag. It is drawn in white because `link="white"` is also in the `<window>` tag.
- The attribute `loop="true"` in the `<window>` tag means the text loops around and comes back in from the right side of the window as soon as the last character of the text has moved completely out of the window. It is not necessary to specify this attribute explicitly, because in tickertape windows `loop="true"` is the default.
- The `<br/>` tag that comes before the first text item forces the text that follows to start just past the window's right edge. Any break or paragraph tag inside tickertape text causes the text that follows to start at the right edge. If the `<br/>` tag were absent, the data would appear starting at the window's left edge.

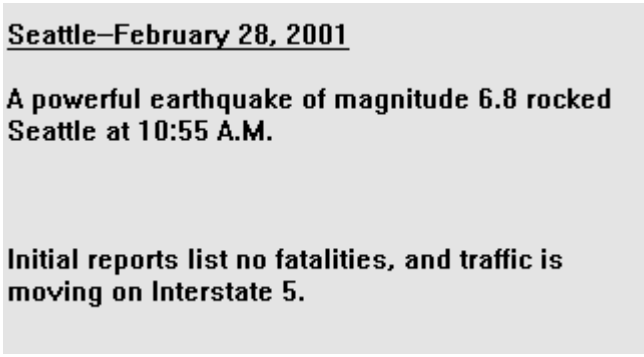
Scrolling News Window

The following sample RealText markup creates a scrolling news window:

```
<window type="scrollingnews" width="240" height="180" scrollrate="20"
duration="25" bgcolor="#4488DD">
<p></p><p></p><p></p>
<font face="System"><b><u>Seattle--February 28, 2001</u></b>
<p>A powerful earthquake of magnitude 6.8 rocked Seattle at 10:55 A.M.</p>
<p>Initial reports list no fatalities, and traffic is moving on Interstate 5.</p>
```

```
<p>Some damage has occurred to buildings in the historic Pioneer Square area.</p>
<p>Seattle mayor Paul Schell is expected to announce a press conference.</p>
</font>
</window>
```

The following figure shows the RealText window at nine seconds into the presentation:


Seattle–February 28, 2001
A powerful earthquake of magnitude 6.8 rocked Seattle at 10:55 A.M.
Initial reports list no fatalities, and traffic is moving on Interstate 5.

Tip: Text in a scrolling news window normally starts at the top of the window and scrolls up. As this sample shows, you can precede the text with tags to push the first line of text to the bottom of the screen.

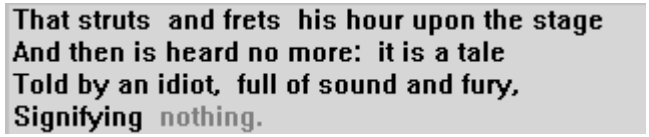
Teleprompter Window

The following example demonstrates a TelePrompter window. This is the RealText (.rt) source file:

```
<window type="teleprompter" height="64" duration="25"
bgcolor="#D2F8B4" extraspaces="ignore" wordwrap="false">
<font face="system">
Out, out, brief candle!
<br/><time begin="3.5"/>Life's but a walking shadow, a poor player
<br/><time begin="7"/>That struts
<time begin="8"/>and frets
<time begin="9"/>his hour upon the stage
<br/><time begin="12"/>And then is heard no more:
<time begin="15"/>it is a tale
<br/><time begin="16"/>Told by an idiot,
```

```
<time begin="17.5"/>full of sound and fury,
<br/><time begin="20"/>Signifying
<time begin="22"/><font color="red">nothing.</font></font>
</window>
```

When the window fills with text and a new line appears, all lines scroll up to make room for the new line. The following illustrates the window when the presentation ends:


That struts and frets his hour upon the stage
And then is heard no more: it is a tale
Told by an idiot, full of sound and fury,
Signifying nothing.

Note the following about TelePrompter windows:

- The wordwrap attribute can be true or false.
- The scrollrate and crawlrate attributes are ignored.
- You can use a <clear/> tag to clear the window and start the next line at the window's upper, left-hand corner.
- Use <time begin/> tags at the start of each line and do not let word wrapping cause too many line breaks between <time/> tags.
- Multiple lines of text with the same begin time cause the preceding text to move up until all new lines appear at the bottom of the window.

PART

III

LEARNING SMIL

The heart of streaming media, SMIL is powerful, but easy to learn. Start with Chapter 7, which covers the uses and structure of a SMIL file, to begin mastering the basics of SMIL. Chapter 8 explains how to incorporate your clips into a presentation, delving into various network protocols such as RTSP and HTTP.

CHAPTER

7

SMIL BASICS

When your streaming presentation contains multiple clips—such as a video and streaming text played together—you use Synchronized Multimedia Integration Language (SMIL) to coordinate the parts. Pronounced "smile," SMIL is a simple but powerful markup language for specifying how and when clips play. This chapter introduces you to SMIL, its advantages, and its syntax rules.

> **For More Information:** Once you are familiar with SMIL, you can refer to "Appendix C: SMIL Tag Reference" beginning on page 453 when you write your SMIL files.

Understanding SMIL

Recommended by the World Wide Web Consortium (W3C), SMIL is designed to be the standard markup language for timing and controlling streaming media clips. SMIL works for a media player similar to the way that HTML works for a Web browser. And just as HTML markup displays in any browser, the standardized SMIL language fosters interoperability between media players. You can find the official SMIL 2.0 specification at the W3C Web site:

http://www.w3.org/TR/smil20/

> **For More Information:** To learn more about multiplayer support, read "Interoperability Between SMIL-Based Players" on page 132.

Advantages of Using SMIL

The following points explain a few of the major advantages of using SMIL:

- Use clips in different locations.

  Because a SMIL file lists a separate URL for each clip, you can put together presentations using clips stored on any server. You can use a video clip on

a RealServer, for example, and an image clip on a Web server. Using SMIL eliminates the need to merge multiple clips into a single streaming file.

- Lay out a presentation.

  When your presentation includes multiple clips, such as a RealVideo clip playing simultaneously with subtitles written in RealText, you use SMIL to arrange the various clips.

- Time and control a presentation.

  SMIL provides powerful timing features that let you easily manage your presentation's timeline. You can keep clips rigidly synchronized, for example, or start an audio clip playing at 2.5 seconds into its internal timeline without changing the encoded clip.

- Layer transparent clips.

  Using RealNetworks' extensions to SMIL, you can easily add transparency to clips and stack them on top of each other. You can turn an opaque graphic into a semi-transparent logo that hovers over a video, for example.

- Create interactive multimedia experiences.

  Using SMIL's advanced features, you can easily create interactive media presentations, such as an audio or video jukebox that plays a different clip each time the viewer clicks a button.

- Link to Web pages.

  SMIL's extensive hyperlinking capabilities allow you to link a streaming presentation to other streaming clips, or to Web pages. Web pages can display automatically at any time during the presentation, or may load only when the viewer clicks a link.

- Stream different presentations to different audiences.

  SMIL lets you stream different clips to different audiences based on criteria such as language preference or available bandwidth. This lets you create multiple presentations, but still have just one link on your Web page. When a viewer clicks that link, the viewer's RealONE Player reads the options in the SMIL file and chooses the appropriate presentation.

> **Note:** RealSystem's SureStream technology also lets you support multiple bandwidth connections within a single clip. For more information, see "SureStream RealAudio and RealVideo" on page 35.

- Display special effects.

  Using SMIL's transition effects and animations, you can create special effects, such as fading one clip into another, or moving a clip around the screen. This lets you duplicate special effects found in advanced video editing programs without making any changes to your streaming clips.

- Put together customized presentations.

  Because a SMIL file is a simple text file, you can generate it automatically for each visitor. You can therefore create different presentation parts, assembling a customized SMIL file for each visitor.

- Include ads in presentations.

  You can use SMIL along with RealServer's advertising extension to insert ads into your presentation. RealServer can deliver image banner ads, as well as media ads in formats such as RealVideo and Flash.

SMIL 1.0 and SMIL 2.0

SMIL 1.0 debuted in 1998. SMIL 2.0, introduced in 2001, updates and expands the SMIL 1.0 capabilities. RealONE Player can play SMIL 1.0 files and SMIL 2.0 files. RealPlayer G2, RealPlayer 7, and RealPlayer 8 can play only SMIL 1.0 files, though. If these older RealPlayers encounter a SMIL 2.0 file, they autoupdate to RealONE Player before displaying the presentation.

> **Note:** This production guide describes SMIL 2.0 only. For information on SMIL 1.0, see *RealSystem iQ Production Guide* for Release 8.

SMIL 2.0 Modules

SMIL defines a number of functional areas, such as timing and hyperlinking. Each functional area breaks down into one or more modules. In turn, each module defines certain attributes and values. The following table lists all the SMIL 2.0 modules, and indicates whether RealONE Player supports them. You may find this information useful if you are familiar with the SMIL 2.0

specification. You do not need to know this information to create a SMIL 2.0 presentation that plays in RealONE Player, however.

SMIL 2.0 Supported Modules

Functional Area	Module	Supported?	Reference
Timing	AccessKeyTiming	**yes**	page 271
	BasicInlineTiming	**yes**	page 240
	BasicTimeContainers	**yes**	page 175
	EventTiming	**yes**	page 262
	ExclTimeContainers	**yes**	page 186
	FillDefault	**yes**	page 258
	MediaMarkerTiming	**yes**	page 274
	MinMaxTiming	**yes**	page 274
	MultiArcTiming	**yes**	page 266
	RepeatTiming	**yes**	page 247
	RepeatValueTiming	**yes**	page 268
	RestartDefault	**yes**	page 274
	RestartTiming	**yes**	page 274
	SyncbaseTiming	**yes**	page 266
	SyncBehavior	**yes**	page 178
	SyncBehaviorDefault	**yes**	page 182
	SyncMaster	no	n/a
	TimeContainerAttributes	**yes**	page 251
	WallclockTiming	**yes**	page 274
Time Manipulations	TimeManipulations	**yes** (animations only)	page 360
Animation	BasicAnimation	**yes**	page 341
	SplineAnimation	no	n/a
Content Control	BasicContentControl	**yes**	page 363
	CustomTestAttributes	no	n/a
	PrefetchControl	**yes**	page 383
	SkipContentControl	**yes**	tbd

(Table Page 1 of 2)

SMIL 2.0 Supported Modules (continued)

Functional Area	Module	Supported?	Reference
Layout	AudioLayout	**yes**	page 218
	BasicLayout	**yes**	page 195
	HierarchicalLayout	**yes**	page 221
	MultiWindowLayout	**yes**	page 204
Linking	BasicLinking	**yes**	page 279
	LinkingAttributes	**yes**	page 290
	ObjectLinking	no	n/a
Media Objects	BasicMedia	**yes**	page 145
	BrushMedia	**yes**	page 148
	MediaAccessibility	**yes**	page 169
	MediaClipping	**yes**	page 242
	MediaClipMarkers	**yes**	page 274
	MediaDescription	**yes**	page 168
	MediaParam	**yes**	page 146
Metainformation	Metainformation	**yes**	tbd
Structure	Structure	**yes**	page 134
Transitions	BasicTransitions	**yes**	page 315
	InlineTransitions	no	n/a
	TransitionModifiers	**yes**	page 330

(Table Page 2 of 2)

SMIL 2.0 Profiles

SMIL also defines *profiles*, which are collections of modules that an application can support. RealONE Player supports the SMIL 2.0 Language Profile, which incorporates most of the SMIL modules listed in the preceding section. The other main profile is the SMIL 2.0 Basic Profile, which is designed primarily for smaller devices, such as mobile phones and portable disc players. The basic profile requires support for only the following modules:

- BasicContentControl
- BasicInlineTiming
- BasicLayout

- BasicLinking
- BasicMedia
- BasicTimeContainers
- MinMaxTiming
- RepeatTiming
- SkipContentControl

Interoperability Between SMIL-Based Players

Because SMIL is an standard markup language, any media player can adopt SMIL as its means for coordinating media clips. Although this allows interoperability between SMIL-based media players, it does not automatically mean that every presentation created for RealONE Player can play in other SMIL-based media players, and vice versa. The following sections explain differences in SMIL presentations that may prevent them from playing in all SMIL-based players.

SMIL Version

SMIL 1.0 and SMIL 2.0 differ significantly. Although most media players that support SMIL 2.0 (including RealONE Player) can also play SMIL 1.0 files, media players that support only SMIL 1.0 (including RealPlayer G2, RealPlayer 7, and RealPlayer 8) cannot play SMIL 2.0 content. Be sure you know whether your target media players support SMIL 1.0, SMIL 2.0, or both.

SMIL Profile

As described in "SMIL 2.0 Profiles" on page 131, a SMIL-based media player can support different SMIL profiles. A media player that supports only the smaller module set of the SMIL 2.0 Basic Profile will not handle all of the attributes defined in the more robust SMIL 2.0 Language Profile. Hence, a presentation developed for RealONE Player may not play to its full capacity in a player based on the SMIL 2.0 Basic Profile. That player should just ignore the SMIL attributes it does not support, however.

Clip Support

SMIL binds different types of clips together, and each SMIL-based media player must also be able to play the presentation's clips, regardless of the player's support for SMIL. For example, RealAudio and RealVideo clips are

proprietary formats that play only in RealPlayer or RealONE Player, whereas QuickTime clips play only in the QuickTime Player. For interoperability, you must stream clips that all of your various target media players can play.

> **Tip:** You can tell if a media player supports a certain type of clip by opening the clip in the player. If the clip plays, you can stream that type of clip to that player.

Media Player Launch Methods

Viewers typically launch streaming media presentations through a Web page hyperlink configured to start a specific player. For example, Web pages that launch RealONE Player link to a Ram file (extension `.ram`), rather than to a SMIL file. If you link directly to the SMIL file, the application registered with the browser to handle the file extension `.smil` launches and attempts to play the presentation. This is not recommended, however, because the launched application may not be one of your target media players.

> **Tip:** RealNetworks recommends that, even with a single SMIL file that plays in multiple media players, you create a separate Web page hyperlink to launch each of your target players. Your viewers can then decide which player they want to use.

> **For More Information:** For more starting RealONE Player with a Web page hyperlink, refer to "Launching RealONE Player with a Ram File" on page 420.

Creating a SMIL File

This section explains the basics of SMIL markup, introducing you to the rules you need to follow when creating a SMIL presentation. If you are familiar with other Web-based markup languages, such as HTML, you will pick up SMIL quickly. You need to be careful, though, because SMIL is less forgiving than HTML. Lapses that may not matter in HTML markup, such as missing quotation marks, missing slashes, or missing end tags, will prevent a SMIL file from working properly.

> **Tip:** You can write a SMIL file with any text editor that can save the file as plain text. Save the file with the file extension `.smil`. Do not include spaces in the file name.

The SMIL 2.0 Tag and Namespace

Rule 1: To create SMIL 2.0 files as described in this manual, the <smil> tag must include the XML namespace for SMIL 2.0.

A SMIL file starts with a <smil> tag and ends with a </smil> tag. If the opening tag is just <smil>, the file is SMIL 1.0:

```
<smil>
  ...SMIL 1.0 markup...
</smil>
```

To create a SMIL 2.0 file and use all the SMIL features described in this manual, the <smil> tag must look like the following:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  ...SMIL 2.0 markup...
</smil>
```

SMIL is based on Extensible Markup Language (XML), which provides the means for defining any number of standard or customized markup languages. The xmlns attribute shown above defines an *XML namespace*. This namespace has just one purpose: to tell RealONE Player that the file is SMIL 2.0 rather than SMIL 1.0. The namespace identifier is in the form of a URL only to ensure uniqueness. RealONE Player does not contact the URL.

Header and Body Sections

Rule 2: A SMIL body section is required, but the header section is optional.

Between the <smil> and </smil> tags, a SMIL file breaks down into two basic subsections: the header and the body. The header is defined between <head> and </head> tags, while the body section falls within <body> and </body> tags, as shown here:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <head>
    ...optional section with all header markup...
  </head>
  <body>
    ...required section with all body markup...
  </body>
</smil>
```

The optional header section is used to give presentation information, to create the layout, and to define features that are used repeatedly. To include a fade-to-black transition effect in your presentation, for example, you first define

the transition type in the header. You can think of the header as defining your presentation's *form*.

The header section is optional because it's not needed for very simple SMIL files. The following SMIL presentation, for example, simply plays three audio clips in sequence. Although the presentation could have a header section that provides presentation information, it doesn't need a layout or any other features that must be defined in the header:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <body>
    <audio src="rtsp://realserver.example.com/one.rm"/>
    <audio src="rtsp://realserver.example.com/two.rm"/>
    <audio src="rtsp://realserver.example.com/three.rm"/>
  </body>
</smil>
```

Within the required body section, you list the clips that you want to play, creating your presentation timeline in the process. Within the body section, you apply the features you defined in the header. For instance, you apply a fade-to-black transition defined in the header to clips listed in the body. You can think of the body as defining your presentation's *content* and *timeline*.

The header and body may each have their own subsections. The header may have a layout section defined between `<layout>` and `</layout>` tags, for example, while the body section uses `<par>` and `</par>` tags to define clips that play together. Other chapters in this guide describe the tags that you can use within the header and body sections.

Tags, Attributes, and Values

Both the header and body of a SMIL file contain tags that have the following form:

<tag attribute="value"/>

Aside from the angle brackets and a possible closing slash, there are three basic parts to a SMIL tag:

tag — The tag name comes just after a left angle bracket. Some tags may consist of just the name, as in the `<body>` tag. Other tags may have attributes.

attribute	Each attribute defines one aspect of the tag. If a tag has several attributes, the order of attributes doesn't matter.
"value"	Most SMIL attributes include an equals sign (=) followed by a value in double quotation marks. In some cases, you choose from a list of predefined values. In other cases, you define your own value. Values may be integers, percentages, names, and so on, depending on what type of value is appropriate for the attribute.

Rule 3: Lowercase or camel case text is required for most tags and attributes.

SMIL tags and attributes must be lowercase. When an attribute or predefined value consists of a compound word, the first letter of all words after the first word is generally capitalized, as in `soundLevel` or `whenNotActive`. This is referred to as "camel case."

A few attributes, such as `root-layout`, are hyphenated. These attributes carry over from SMIL 1.0. They have been kept the same for consistency. Some new SMIL 2.0 attributes, such as `accesskey`, are meant to be compatible with HTML 4.0 and, in accordance with the HTML 4.0 specification, do not capitalize letters in compound words.

Rule 4: Attribute values must be enclosed in double quotation marks.

Attribute values, such as `video_region` in `region="video_region"`, must be enclosed in double quotation marks. Do not add any blank spaces between the quotation marks and the value they enclose.

Rule 5: File names and paths must observe letter cases.

In clip source tags, paths and file names can be uppercase, lowercase, or mixed case. All of the following path and file name examples are allowable, for example:

```
<audio src="rtsp://realserver.example.com/song.rm"/>
<audio src="rtsp://RealServer.Example.com/Song.rm"/>
<audio src="rtsp://realserver.example.com/SONG.rm"/>
```

However, the path and file name in the tag should match the clip's path and file name exactly as it appears on the server computer's operating system. For instance, the following clip source tag may not work if the clip is actually named `SONG.rm`:

```
<audio src="rtsp://realserver.example.com/song.rm"/>
```

Binary and Unary Tags

Rule 6: All tags must have an end tag or close with a forward slash.

Some SMIL tags, called *binary* tags, have a corresponding end tag. For example, the <body> tag has the end tag </body>. When a tag has no corresponding end tag, it is called a *unary* tag, and it must close with a forward slash as shown in this example:

```
<audio src="first.rm"/>
```

> **Warning!** Omitting a closing slash where it's needed, or adding it where it's not required is one of the easiest ways to create an error in a SMIL file. Take care always to include a closing slash with a unary tag, and to leave it out of the first tag in a binary pair.

Changing a Unary Tag to a Binary Tag

Several SMIL tags can be either binary or unary, depending on how they operate. For example, a unary <video/> tag plays a video clip:

```
<video ...specifies a video to play, and closes with a forward slash... />
```

However, you can also include a hyperlink with <video/> tag to link the clip to another clip or a Web page. To do this, you change the <video/> tag from unary to binary so that it can enclose an <area/> tag, as shown here:

```
<video ...specifies a video to play, and uses an end tag... >
  <area ...defines an image map, and closes with a forward slash... />
</video>
```

This manual tells you which tags can be both unary and binary, and explains the circumstances under which you use the unary or binary version.

SMIL Recommendations

Although not strict rules, the following recommendations will help you keep your SMIL markup organized and understandable.

Recommendation 1: Use HTML-style comments to annotate your SMIL file.

As in HTML, SMIL has a comment tag that starts with these characters:

```
<!--
```

and ends with these characters:

```
-->
```

The ending does not include a forward slash:

```
<!-- This is a comment -->
```

A comment can be any number of lines long. It can start and end anywhere in a SMIL file. Multiple comments cannot be nested, though. Use comments to describe what various sections of your SMIL presentation are meant to do. This helps other people understand your presentation more easily.

Recommendation 2: Use indentation to clarify how your SMIL file is organized.

Although indenting SMIL markup is not required, it helps you to keep track of the SMIL file's structure. You typically indent markup by pressing the **Tab** key once for each level of indentation. In a clip group, for example, the group tags are indented one level from the body tags, and the clip tags are indented one level from the group tags, as shown here:

```
<body>
 <seq>
  <audio src="rtsp://realserver.example.com/one.rm"/>
  <audio src="rtsp://realserver.example.com/two.rm"/>
 </seq>
</body>
```

SMIL Tag ID Values

Any SMIL tag can have an ID in the form `id="value"` that allow other SMIL elements to identify them. Some SMIL tags require IDs. For example, each layout region requires an ID that you use to assign clips to play in the region. For other tags, IDs are optional depending on whether another SMIL element interacts with that tag. The following are rules and suggestions that apply to the IDs of any SMIL tag:

- All IDs for all tags in a SMIL file must be unique. If you define several `<region/>` tags, for example, each tag must have a unique ID. No `<region/>` tag can have the same ID as a `<transition/>` tag or a `<video/>` tag, for instance.
- As with all SMIL values, IDs are case-sensitive. The attributes `id="videoregion"` and `id="videoRegion"` are different, for example. It is a good idea to follow a consistent practice, such as always making IDs lowercase.

- Do not use words separated by spaces in an ID. If you use two or more words for an ID, combine the words, or separate the words with an underscore or hyphen, as in `videoregion`, `video-region`, or `video_region`.
- The first character for an ID can be a letter, a colon, or an underscore. It cannot be a number or a special character such as an ampersand. You can use numbers and special characters after the first character, however. For example, you can use `id="video3"` as an ID, but not `id="3video"`.
- There is no minimum or maximum length for IDs.
- You may find it convenient to adopt a system for specifying IDs. You might use the suffix `_region` for all region IDs, for example, or a `transition_` prefix for all transition effect IDs.

Using Customized SMIL Attributes

SMIL can be customized, and RealNetworks has developed many extensions to SMIL 2.0 functionality. SMIL regulates how customizations can be added, though, to avoid potential conflicts between different media players. A customized attribute always has a prefix, and takes the following form:

prefix:*attribute*="*value*"

The prefix is user-defined, but the attribute name is always predefined. The following is an example of RealNetworks' `backgroundOpacity` attribute, using a prefix of `rn`:

```
rn:backgroundOpacity="50%"
```

When RealONE Player encounters this tag, it recognizes that `backgroundOpacity` is a valid attribute, but not a standard SMIL attribute. It uses the `rn` prefix to match the attribute to a namespace declared in the `<smil>` tag. The namespace must therefore use the same user-defined prefix as the attribute. You can add the additional namespace to the `<smil>` tag after the SMIL 2.0 namespace:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language"
xmlns:rn="http://features.real.com/2001/SMIL20/Extensions">
```

If RealONE Player recognizes the namespace, it knows how to handle the customized attribute. This allows RealONE Player to support any number of customized attributes developed by RealNetworks or other parties.

RealNetworks Extensions Namespace

RealNetworks has created many customized attributes that you can use in SMIL files played in RealONE Player. To use these attributes, you must declare the following namespace in the <smil> tag:

```
xmlns:rn="http://features.real.com/2001/SMIL20/Extensions"
```

This manual always uses rn: as the attribute prefix for RealNetworks extensions. If you decide to use a different prefix, it's best to use a short, single word, or just a few letters.

A Closer Look at Namespaces

Namespaces and prefixes for customized attributes are not hard to declare and use, but they can be confusing at first if you are not familiar with XML. The following sections delve more deeply into namespaces and their associated prefixes for those who want a better understanding of this issue. When in doubt, though, just follow the examples in this guide, using the given prefixes when defining a namespace and a custom attribute.

Why does SMIL use namespaces?

Each customized attribute is defined in conjunction with a unique namespace so that SMIL-based media players can use different attributes that happen to have the same name. An attribute named find might perform one function when defined with one namespace, and a different function when defined with another namespace. This allows different parties to create customized SMIL attributes without being concerned about duplicate attribute names.

Why are prefixes used?

A prefix ties an attribute to a namespace. Consider the example of two different find attributes in the same SMIL file. When RealONE Player has to interpret what a particular find attribute does, it matches the attribute to its namespace through the prefix. If there were no prefix, RealONE Player would not know which namespace goes with which attribute.

Why are prefixes user-definable?

If the parties who developed custom attributes also defined specific prefixes, there could be duplicate attribute names and prefixes that RealONE Player could not resolve. Suppose that two parties developed two new SMIL attributes, both called fd:find, but each defined against a different namespace.

If you used both fd:find attributes in your presentation, RealONE Player would not know which attribute goes with which namespace.

Because prefixes are user-definable, though, you could change the prefix for one of the attributes, making it xy:find, for example. You would then use the same xy prefix in the associated namespace so that RealONE Player could match each find attribute to its namespace. This provides flexibility for parties developing customized attributes, but it also places responsibility on the SMIL author to match customized attributes to namespaces through prefixes.

Tips for Defining Namespaces

- To summarize, there are three required parts of a customized attribute:
  - a user-defined attribute prefix such as rn:
  - a predefined attribute and value pair that uses the prefix, such as rn:backgroundOpacity="50%"
  - a predefined namespace that includes the user-defined prefix. The attribute is always defined against a namespace, such as the RealNetworks extensions namespace:

    xmlns:rn="http://features.real.com/2001/SMIL20/Extensions"
- Within the <smil> tag, you can declare each namespace on a separate line for easier reading. SMIL ignores extra spaces, carriage returns, line breaks, and tabs used simply to align text in a file. Just make sure that the closing angle bracket of the <smil> tag appears after the last namespace.
- It's OK to declare a namespace in the <smil> tag even if you don't use any customized attributes associated with that namespace.
- Support for a customized attribute must be built into a media player. Other SMIL-based players may not support the same customized attributes as RealONE Player, and vice versa. But if a SMIL-based media player does not support a customized attribute, it simply ignores the attribute.

Viewing SMIL Source Markup

RealONE Player has a **View>Clip Source** command that shows the SMIL markup of the current presentation. Using this command is a good way to learn how a SMIL presentation is put together. The RealServer or Web server

hosting the presentation sends the markup as an HTML page that opens in your default Web browser.

Access to SMIL source information is denied for secure presentations that require a user name and password. The RealServer administrator may also disallow access to the SMIL source file, or allow access to the source file but conceal the full paths of clips. When access is allowed, the Web page showing the SMIL syntax includes a hypertext link for each clip in the presentation. Clicking a link displays a new Web page with information about the corresponding clip, including its size, buffer time, and streaming bit rate.

Playback Differences from SMIL 1.0

If you have created SMIL 1.0 presentations for playback in RealPlayer G2, RealPlayer 7, or RealPlayer 8, this section will help bring you up-to-date with changes in SMIL 2.0.

Behavioral Changes

The SMIL 2.0 specification requires changes to RealONE Player's handling of some basic features that carry over from SMIL 1.0:

- RealONE Player now treats clips without internal timelines, such as images, as having an intrinsic duration of 0 seconds. This means you must include a `dur` or `end` attribute to make these clips display at all. For information on durations, see "Setting Durations" on page 243.
- A clip without a `fill` attribute defaults to `fill="auto"`, which can be equivalent to `fill="remove"` or `fill="freeze"` depending on the circumstance. See "Setting a Fill" on page 251.
- In a `<par>` group, a `fill="freeze"` attribute displays a clip only until the group ends. If the presentation ends when the group ends, the clip does not stay frozen on the screen as it did in SMIL 1.0. Instead, it is removed once the group is no longer active. To display a clip after its group ends, use `fill="hold"` and `erase="never"` in the clip tag. For more information on these new attributes and values, see "Displaying a Clip Throughout a Presentation" on page 255.

Updating SMIL 1.0 Files to SMIL 2.0

A SMIL 1.0 presentation created for an earlier version of RealPlayer will play in RealONE Player. If you want to update a SMIL 1.0 presentation to SMIL 2.0, however, you have to change the <smil> 1.0 tag to a SMIL 2.0 tag:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
```

The following table provides a quick reference for changing other SMIL 1.0 tags and attributes to their SMIL 2.0 equivalents. Once you make these changes, you can add any other SMIL 2.0 features to your presentation. Your SMIL file will play only in RealONE Player, however.

Tag and Attribute Changes from SMIL 1.0 to SMIL 2.0

SMIL 1.0 Element	SMIL 2.0 Tag or Attribute	Reference
Layout Tags and Attributes		
background-color	backgroundColor	page 215
Clip Source Tags and Attributes		
?bitrate=*nnnn*	<param name="bitrate" value="*nnnn*"/>	page 146
?reliable=true	<param name="reliable" value="true"/>	page 148
?bgcolor=*RRGGBB*	<param name="bgcolor" value="*RRGGBB*"/>	page 162
Timing Tags and Attributes		
repeat	repeatCount	page 247
clip-begin	clipBegin	page 242
clip-end	clipEnd	page 242
Hyperlinking Tags and Attributes		
<anchor/>	<area/>	page 283
show="new"	external="true" sourcePlaystate="play"	page 294
show="pause"	external="true" sourcePlaystate="pause"	page 294
target="*ID*"	*URL#ID*	page 304
Switch Tag Attributes		
system-bitrate	systemBitrate	page 370
system-language	systemLanguage	page 367
system-captions	systemCaptions	page 372

CHAPTER

8

CLIP SOURCE TAGS

For every clip you play in your presentation, such as an audio clip, video clip, or text clip, you add a source tag to your SMIL file. This chapter explains the basics of clip source tags, explaining how to write URLs that tell RealONE Player where to find clips. It also tells how to modify certain characteristics, such as background transparency, when clips play.

Creating Clip Source Tags

Each time you want a clip to appear in a presentation, you write a clip source tag that tells RealONE Player where to find the clip. The source tag URL may point RealONE Player to a clip on RealServer, a Web server, or even the viewer's local computer. A typical clip tag looks like this:

```
<audio src="rtsp://realserver.example.com:554/audio/song1.rm"/>
```

Within each clip source tag, a src attribute lists the clip location. The section "Writing Clip Source URLs" on page 151 explains how to specify a URL with the src attribute. As described in subsequent chapters, clip source tags can also contain other attributes that control clip timing and layout. The following table lists the different clip source tags you can use in a presentation.

Clip Source Tags

Clip Tag	Used For
<animation/>	animation clips such as a Flash Player file (.swf)
<audio/>	audio clips such as RealAudio (.rm)
<brush/>	color block used in place of a clip (See "Creating a Brush Object" on page 148.)
<img/>	JPEG (.jpg), GIF (.gif), or PNG images (.png) (See "Setting a Clip's Streaming Speed" on page 146.)
<ref/>	miscellaneous clip type, such as RealPix (.rp) or Ram (.ram) file

(Table Page 1 of 2)

Clip Source Tags (continued)

Clip Tag	Used For
<text/>	static text clips (.txt)
<textstream/>	streaming RealText clips (.rt)
<video/>	video clips such as RealVideo (.rm)

(Table Page 2 of 2)

The particular clip source tag you use does not affect clip playback because RealONE Player determines the actual clip type by other means. Specifying a video clip with an <audio/> tag, for example, does not prevent RealONE Player from recognizing that the clip contains video. Although using a tag appropriate to the clip's contents helps you keep track of clips, you could specify all clips with <ref/> tags, for example. Other clip tags cannot be used in place of the <brush/> tag, however.

Adding a Clip ID

RealNetworks recommends that every clip source tag include a user-defined ID in the form id="*ID*". Clip IDs are not always necessary, but you will need to use them when building complex presentations in which other SMIL elements refer to clips. Clicking a hyperlink, for example, can start a clip playing. In this case, the hyperlink uses the clip's ID to identify which clip to start. RealONE Player never displays IDs onscreen. Here is an example of a clip ID:

```
<video src="video1.rm" id="video1"/>
```

> **For More Information:** For information about selecting ID values, see "SMIL Tag ID Values" on page 138.

Setting a Clip's Streaming Speed

Clips such as audio, video, and animation have a streaming speed set by the tools used to encode or tune the clips. For these clips, never use SMIL to set a streaming speed. For static clips such as images (GIF, JPEG, or PNG) and text (static text files and streaming RealText files), however, you can use SMIL to change the clip's streaming bandwidth from the default of 12 Kilobits per second (approximately 12000 bits per second). This works only when streaming from RealServer. With Web server hosting, there is no way to set a static clip's streaming speed.

> **Tip:** Small text files stream so quickly that they rarely interfere with other clips. Therefore, you generally do not need to set the

streaming bandwidth for text files. You should set a streaming bandwidth for image files larger than 5 Kilobytes, if the 12 Kbps default value is too high or too low for your target audience.

For More Information: For background information on streaming speeds, see "Audience Bandwidth Targets" on page 32 and "Clip Bandwidth Characteristics" on page 33.

Using the bitrate Parameter

To set a static clip's streaming speed, you modify the clip source tag to use binary tags, as described in the section "Binary and Unary Tags" on page 137. Within the binary clip tag, you add a `<param/>` tag with the name `bitrate`, and specify the speed in bits per second. The following example sets an image to stream at approximately 5 Kilobits per second:

```
<img src="poster.jpg" region="image_region" dur="15s">
  <param name="bitrate" value="5000"/>
</img>
```

RealONE Player does not display an image clip until it has received all the clip's data, and the clip is scheduled to display according to the SMIL timeline. For clips that have no intrinsic duration, such as images and text files (though not RealText clips), you must specify a duration.

For More Information: For more on image durations, see "Setting Durations" on page 243.

Example of Streaming Images Slowly

Using the `bitrate` parameter, you can set a high streaming speed for a clip to take advantage of available bandwidth and stream the image quickly. Or you can set a low bit rate to ensure that streaming the image does not interfere with playing another clip at the same time. The following example shows three sequential image files set to stream at 1000 Kbps to ensure that a video playing in parallel does not stall:

```
<par>
  <video src="video.rm" region="video_region"/>
  <par>
    <seq>
      <img src="ad1.gif" begin="30s" dur="60s" region="ad_region">
        <param name="bitrate" value="1000"/>
      </img>
```

```
    <img src="ad2.gif" dur="120s" region="ad_region">
      <param name="bitrate" value="1000"/>
    </img>
    <img src="ad3.gif" dur="60s" region="ad_region">
      <param name="bitrate" value="1000"/>
    </img>
   </seq>
  </par>
</par>
```

For More Information: For more on SMIL timing, see Chapter 12. Chapter 10 explains group tags such as <par> and <seq>.

Ensuring Reliable Clip Transmission

You can use the reliable value in a <param/> tag to indicate that a clip must be delivered to RealONE Player under any circumstances. During extremely adverse network conditions, RealSystem will halt the presentation if necessary rather than drop the clip. The following example shows the reliable parameter set for an image:

```
<img src="poster.jpg">
  <param name="bitrate" value="5000"/>
  <param name="reliable" value="true"/>
</img>
```

Tip: Use the reliable parameter sparingly, and only for small, important elements of your presentation. Even without this parameter, RealSystem generally ensures that very little data loss occurs in transmission.

Warning! The reliable parameter is not for use with large clips such as videos. These clips are designed to play well even if some data is lost in transmission. Using the reliable parameter with these clips may cause your presentation to stall.

Creating a Brush Object

The <brush/> tag lets you create a colored area that displays in a region. It functions just like a clip source tag. For example, you can control when the brush object appears by using SMIL timing commands, and you can change its size and color with SMIL animation tags. Because it does not link to an external clip, though, the <brush/> tag does not use a src attribute. Instead, it uses a color attribute to define the color used:

```
<brush color="blue" region="region_1" dur="5s"/>
```

Black is the default color for a brush object. To specify a different color, use a predefined color name, a hexadecimal color value, or an RGB value.

> **Tip:** Appendix D explains the types of color values that you can use with SMIL color attributes.

Using a Ram File as a Source

A Ram file (.ram) is typically used to launch RealONE Player and give it the URL of the clip or SMIL presentation to play. But you can also use a Ram file as a source of content within a SMIL file. Because a Ram file can list several clips in sequence, you may find it useful to specify a Ram file (a different Ram file from the one used to launch the presentation) within your SMIL file.

To illustrate how a Ram file is useful, suppose that your main SMIL presentation defines an online radio application that plays a preset song list that changes daily. You could list all the songs within the SMIL file in a sequence, like this:

```
<seq>
  <audio src="song1.rm"/>
  <audio src="song2.rm"/>
  ...more songs...
</seq>
```

Each day, though, you'd need to modify your main SMIL file to update the playlist. It's easier in this case to have the SMIL file request a Ram file through a <ref/> tag:

```
<ref src="songs.ram"/>
```

You then modify the Ram file each day with your new playlist. The Ram file simply gives the full URL to each song in the order in which they play:

```
rtsp://realserver.example.com/song1.rm
rtsp://realserver.example.com/song2.rm
...more songs...
```

When you use a Ram file as a source, you can add SMIL timing and layout attributes to the <ref/> tag. In a playlist of videos, for example, you could assign all the videos to play in the same region, which your main SMIL presentation would define. Or you could use timing attributes to give the entire sequence of clips a maximum duration, for instance. You cannot use SMIL attributes within a Ram file, however.

For More Information: For more on Ram files, see "Launching RealONE Player with a Ram File" on page 420. Note that a Ram file can also list other Ram or SMIL files, as well as clips.

Using a SMIL File as a Source

A SMIL file can also use another SMIL file as a source. Simply use a <ref/> tag to refer to the secondary SMIL file:

```
<ref src="rtsp://realserver.example.com/presentation2.smil"/>
```

Unlike a Ram file, a SMIL file can do more than list a simple sequence of clips. A secondary SMIL file can play clips in parallel, for example, and use SMIL timing and layout attributes to organize its clips.

Handling Layouts

When a secondary SMIL file contains visual clips, you assign the entire SMIL file to play within a region defined in the primary SMIL file:

```
<ref src="rtsp://realserver.example.com/presentation2.smil" region="region_1"/>
```

In this case, it's best to define the playback region in the main SMIL file (region_1) to be the same size as the root-layout area of the secondary SMIL file. If the playback region and the secondary SMIL presentation are different sizes, the playback region's fit attribute determines how the SMIL presentation fits the region.

For More Information: For information on defining layouts and assigning clips to play in regions, see Chapter 11.

Using Timing Attributes

Timing attributes in the primary SMIL file can override the timeline of the secondary SMIL file. Suppose that presentation2.smil lasts 10 minutes when played by itself, but you set a 5-minute duration in the <ref/> tag in the main SMIL file:

```
<ref src="rtsp://realserver.example.com/presentation2.smil" dur="5min"/>
```

In this case the duration specified in the main SMIL file cuts off the last half of presentation2.smil.

For More Information: Timing attributes are described in Chapter 12.

Writing Clip Source URLs

Every clip source tag, except for a <brush/> tag, requires an src attribute that provides the URL for the clip. RealONE Player uses this URL to request the clip from a server. The URL you specify varies depending on whether the clip resides on RealServer, a Web server, or the viewer's local machine.

> **Tip:** As you develop a presentation on your computer, use local URLs. Then put in a base URL, or specify full URLs for each clip, when you are ready to stream your presentation. Chapter 20 explains how to move clips to a server and write a Ram file to launch RealONE Player.

Linking to Local Clips

As you develop your presentation, it is easiest to keep your SMIL file and your clips in the same directory on your local computer. Within your SMIL file, the src parameter for each clip source tag can simply give the file name:

```
<audio src="song1.rm"/>
```

Creating Relative Links to Other Directories

RealONE Player can also follow the same relative links that you can use in a Web page. For example, the following src attribute specifies a clip that resides one level below the SMIL file in the audio folder:

```
<audio src="audio/song1.rm"/>
```

The following example specifies a clip that resides one folder level above the SMIL file:

```
<audio src="../song1.rm"/>
```

The next example creates a link to a clip that resides in an audio folder that is at the same level as the folder that contains the SMIL file:

```
<audio src="../audio/song1.rm"/>
```

> **Tip:** You can find additional information about relative directory syntax in an HTML reference guide.

Writing Absolute Links

Alternatively, you can use local, absolute links to specify exact locations. The syntax for absolute links is the same as with HTML. It varies with operating systems, however, and you should be familiar with the directory syntax for the

system you are using. For example, the following absolute link syntax works for Windows computers, but not on Unix or the Macintosh:

```
src="file://c:\audio\first.rm"
```

Creating a Base URL

When you are ready to stream your presentation, you can add a base URL to your SMIL file. This is convenient if all or most of your clips reside on the same server. This preserves the local, relative syntax you used when developing your presentation, readying your presentation for streaming in a single step. You add the base URL to the file in the SMIL header section through a <meta/> tag as shown here:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <head>
    <meta name="base" content="rtsp://realserver.example.com/"/>
    ...layout information...
  </head>
  <body>
    <par>
      <audio src="song1.rm"/>
      <textstream src="lyrics/words1.rt" .../>
      <img src="http://www.example.com/images/album3.gif" .../>
    </par>
  </body>
</smil>
```

Because the third clip in this example uses a full URL, the base target is ignored. RealONE Player requests the image from the specified Web server using the HTTP protocol. For the first two clips, however, the src values are appended to the base target, effectively giving the clips the following URLs:

```
rtsp://realserver.example.com/song1.rm
rtsp://realserver.example.com/lyrics/words1.rt
```

Using a base target is highly recommended. If no target is given, RealONE Player assumes that the clip paths are relative to the location of the SMIL file. In the preceding example, for instance, RealONE Player would look for song1.rm in the same directory that holds the SMIL file, requesting the clip with the same protocol used to request the SMIL file.

Linking to Clips on RealServer

When clips reside on RealServer, use an RTSP URL in the base target. Or, you can specify an RTSP URL in each clip's src attribute. An RTSP URL in a clip source tag looks like this:

```
<audio src="rtsp://realserver.example.com:554/audio/first.rm"/>
```

The following table explains the URL components. Your RealServer administrator can give you the RealServer address, RTSP port, and directory structure.

RealServer URL Components

Component	Specifies
rtsp://	RTSP protocol. Although RealServer also supports HTTP, streaming clips typically use RTSP.
realserver.example.com	RealServer address. This varies with each RealServer. It typically uses an identifier such as realserver instead of www. Or it may use a TCP/IP address (such as 172.2.16.230) instead of a name.
:554	RealServer port for RTSP connections. Port 554 is the default, so you can leave this out of URLs unless the RealServer administrator chose a different port for RTSP communication. If the port number is required, separate it from the address with a colon.
/audio/	RealServer directory that holds the clip. The directory structure may be several levels deep. RealServer also uses "mount points" that invoke certain features, such as password authentication. Because these mount points appear to be directories in the URL, the request path does not mirror the actual directory path on the RealServer computer. The RealServer administrator can tell you the mount points and directories in the path.
first.rm	Clip file name.

For More Information: For more information on RTSP, see "The Difference Between RTSP and HTTP" on page 417.

Linking to Clips on a Web Server

To use a clip hosted on a Web server, use a standard HTTP URL in the base target, or in each clip's src attribute. RealServer also supports the HTTP protocol, but for clips streaming from RealServer, you typically use the RTSP

protocol or the specialized CHTTP protocol, which is described in the following section. An HTTP URL in a clip source tag looks like this:

```
<img src="http://www.example.com/images/logo.gif"/>
```

Warning! Although a Web server can host any clip, a Web server cannot perform all the functions of RealServer. For more information, see "Limitations on Web Server Playback" on page 428.

Note: RealONE Player does not support secure HTTP connections directly. However, e-commerce forms created in Flash can transmit secure HTTP data through the viewer's Web browser. For more information, see "Secure Transactions" on page 87.

Caching Clips on RealONE Player

RealONE Player does not cache files by default, but you can make it cache on disk any files delivered through HTTP. You may want to cache images used in different SMIL presentations that site visitors play. An example is an Internet radio station that uses GIF logos and on-screen buttons. As long as the GIFs reside in the RealONE Player cache, the server does not have to resend the files if, for example, the user clicks a link that opens a new SMIL presentation containing the same images.

Caching works only for files delivered through HTTP. You should not try to cache large clips that would be served better through RTSP, such as video, audio, Flash, and RealPix clips. (RealONE Player caches RealPix images in memory, but not on disk, for the duration of the RealPix presentation.) Nor should you cache ads or images that do not appear repeatedly in your presentation.

Using the CHTTP Caching Protocol

RealONE Player does not cache all items streamed by HTTP. Instead, you designate files to cache by using `chttp://` instead of `http://` in the file URLs. When RealONE Player reads a CHTTP URL in a SMIL file, it first checks its disk cache for the file. If the file is not present, RealONE Player requests the file through HTTP, storing the file in its cache. Because RealONE Player interprets a `chttp://` URL as a special instance of HTTP, caching works for any file stored on an HTTP-compatible server.

If a file is stored in RealONE Player's cache, RealONE Player reuses the file instead of requesting it again from the server, as long as a CHTTP URL is used. The cached version is not used, though, if the URL starts with http:// or differs in any way from the original CHTTP URL. The following SMIL example indicates that the specified GIF image should be downloaded and cached for later use:

```
<img src="chttp://www.example.com/images/image1.gif" .../>
```

Example of Using CHTTP in a Presentation

When caching files, download the cached items before streaming other elements. You can do this by placing the cached elements in a SMIL <seq> group ahead of the streamed elements. In the following example, the two logos quickly download before the RealVideo and RealText clips play. If the visitor plays another presentation that also caches the two images, RealONE Player first checks its cache. If it finds the images, it skips directly to the streaming clips:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
 ...header omitted...
 <body>
  <seq>
   <!-- First, download and cache these two logos. -->
   <img src="chttp://www.example.com/logo1.gif region="logo1" ...>
    <param name="bitrate" value="20000"/>
   </img>
   <img src="chttp://www.example.comlogo2.gif region="logo2" ...>
    <param name="bitrate" value="20000"/>
   </img>
   <par>
    <!--Second, stream these 2 clips in parallel. -->
    <textstream src="rtsp://realserver.example.com/news.rt" region="news" .../>
    <video src="rtsp://realserver.example.com/newsvid.rm" region="video1" .../>
   </par>
  </seq>
 </body>
</smil>
```

Controlling the RealONE Player Cache

Because RealONE Player supports the same HTTP header fields used to control file expiration in Web browser caches, it can carry out caching directives set by Web servers. Thus, you can reuse Web page images in RealONE Player presentations without losing control of how these images are

cached. This section describes how to use HTTP headers to control the RealONE Player cache, and how RealONE Player manages its cache. Documentation for most Web servers includes information about how to set fields in HTTP header files.

Overriding Caching with Cache-Control

The `Cache-Control` command of an HTTP header file can override caching of a RealONE Player file requested through `chttp://`. A file requested through CHTTP is not cached if any of the following are present as meta-information in the HTTP header file:

- `Cache-Control:no-cache`
- `Cache-Control:no-store`
- `Cache-Control:private`
- `Cache-Control:must-revalidate`

Cache Size and Expiration Rules

RealONE Player caches files within its home directory in a folder named `cache_db`. This cache is independent of any Web browser cache. The default RealONE Player cache size is 4 MB. Unless an HTTP header sets a file lifetime, the cached file expires after 4 hours, although a subsequent request for a cached item restarts the item's expiration clock. As the cache fills, RealONE Player begins to delete unexpired items to reclaim needed disk space on a first-in, first-out basis.

> **Note:** RealONE Player users can control some aspects of RealONE Player's cache by disabling the cache, setting the amount of disk space available for the cache, and emptying the cache. Users carry out these actions through the RealONE Player preferences. For more information, see the RealONE Player online help.

Changing the Lifetime of a Cached File

Within an HTTP header, you can have `Cache-Control:max-age` set the "time to live" (TTL) for a cached file, overriding the default expiration time. Expressed in seconds, the maximum age is added to the current time to yield the file's expiration time. This value must be between 60 seconds and one year. For example:

```
Cache-Control:max-age=172800
```

If you do not use the Cache-Control:max-age field, you can have the Expires field determine the file's expiration time. The Expires field takes as an attribute a date string that defines when the cached element expires, relative to the caching computer's clock. The date string is formatted as follows:

Expires= *Wdy, DD Mon YYYY HH:MM:SS* GMT

The weekday is optional. In the following two examples, the first example includes a weekday designation, the second one does not:

Expires= Fri, 17 Mar 2000 19:37:09 GMT

Expires= 17 Mar 2000 19:37:09 GMT

The weekday and month abbreviations are as follows:

Day of week:	Mon, Tue, Wed, Thu, Fri, Sat, Sun
Month:	Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec

Note: The entry is not cached if the value in the Expires: field predates the current date and time.

Modifying Clip Colors

The clip color attributes summarized in the following table are primarily for images in the GIF, JPEG, or PNG format. They can also be used for dynamic clips, though, especially those that include transparency, such as Flash clips. They should not be used for streaming video, however. Note that because these attributes are specific to RealONE Player, other SMIL-based media players may not recognize them.

Clip Streaming and Color Attributes

Attribute	Value	Function	Reference
rn:backgroundOpacity	*percentage*	Adjusts background transparency.	page 158
bgcolor	*nnnnnn*	Substitutes color for transparency.	page 162
rn:chromaKey	*color_value*	Turns selected color transparent.	page 159
rn:chromaKeyOpacity	*percentage*	Adds opacity to chromaKey.	page 159
rn:chromaKeyTolerance	*color_value*	Widens range of chromaKey.	page 160
rn:mediaOpacity	*percentage*	Makes opaque colors transparent.	page 158

For More Information: Appendix D explains the types of color values you can use with SMIL color attributes.

Adjusting Clip Transparency and Opacity

Two customized attributes let you add transparency to all opaque colors in a clip (rn:mediaOpacity), or adjust transparency in just the clip's background color (rn:backgroundOpacity). You can use these attributes separately or together. Using either of these attributes requires that you declare the following namespace in the <smil> tag:

```
xmlns:rn="http://features.real.com/2001/SMIL20/Extensions"
```

> **For More Information:** For the basics of customized namespaces, see "Using Customized SMIL Attributes" on page 139.

Adding Transparency to All Opaque Colors

The attribute rn:mediaOpacity in a clip source tag causes opaque areas in the clip to become transparent. The attribute takes a percentage value in the range from 0% (fully transparent) to 100% (fully opaque). In the following example, the opaque areas of a GIF image are rendered partially transparent, making them blend with a region's background color or an underlying clip:

```
<img src="button.gif" rn:mediaOpacity="50%" .../>
```

> **Note:** If a clip is 50 percent or more transparent (that is, it has a value from 0 to 50 for rn:mediaOpacity), hyperlinks defined for the clip will not work. Clicking the clip will open hyperlinks on clips beneath the partially transparent clip, however. Chapter 14 explains SMIL hyperlinks.

Adjusting Transparency in a Clip's Background Color

Using the rn:backgroundOpacity attribute, you can modify the opacity of a clip's background, making the background color more transparent or more opaque. This attribute works only for clips that designate a specific background color, such as GIF, PNG, or RealText clips. It does not work for clips like JPEG images or RealVideo clips that do not explicitly specify a background color.

The rn:backgroundOpacity attribute takes a percentage value in the range from 0% (fully transparent) to 100% (fully opaque). In the following example, the background color specified in the image's palette, which may be fully opaque or fully transparent, is rendered partially opaque:

```
<img src="button.gif" rn:backgroundOpacity="50%" .../>
```

If the background color is partially transparent already, rn:backgroundOpacity increases the opacity. If a clip's background is 50 percent transparent already,

for example, using rn:backgroundOpacity="50%" adds another 50 percent to the opacity, making the background 75 percent opaque.

> **Tip:** If a clip's background is fully opaque, you can use just rn:mediaOpacity to render the background and all other colors transparent. If the clip's background is partially transparent already, rn:mediaOpacity will not affect the background, and you can use both rn:mediaOpacity and rn:backgroundOpacity in the same clip tag.

Substituting Transparency for a Specific Color

For clips that do not include native transparency, such as JPEG images and Flash clips, you can use three attributes to define a color (rn:chromaKey), or a range of colors (rn:chromaKeyTolerance), that RealONE Player renders transparent or partially transparent (rn:chromaKeyOpacity). Using these attributes requires that you declare the following namespace in the <smil> tag:

```
xmlns:rn="http://features.real.com/2001/SMIL20/Extensions"
```

> **For More Information:** For background on customized attributes, see "Using Customized SMIL Attributes" on page 139.

Selecting a Color to Render Transparent

You can use rn:chromaKey to specify a color that RealONE Player will render transparent. In the following example, the hexadecimal color #808080 is made transparent in a JPEG clip:

```
<img src="picture.jpg" rn:chromaKey="#808080"…/>
```

> **Tip:** You can specify colors by using any color value described in Appendix D. For example, the preceding attribute could use the RGB value "rgb(128,128,128)" instead of the hexadecimal "#808080".

Using Partial Transparency

You can use the rn:chromaKeyOpacity attribute to make the color value selected by rn:chromaKey partially transparent instead of fully transparent. The chromaKeyOpacity attribute uses a percentage value from 0% (the default value of full transparency) to 100% (fully opaque). In the following example, the selected color is rendered 50 percent transparent instead of fully transparent:

```
<img src="picture.jpg" rn:chromaKey="#808080" rn:chromaKeyOpacity="50%"/>
```

Expanding the Transparency Range

To achieve the desired transparency effect, you may need to use the attribute rn:chromaKeyTolerance to widen the range of colors selected by rn:chromaKey. Although rn:chromaKeyTolerance uses a value that looks like a single color designation, the value actually specifies a *range* of colors around (both above and below) the rn:chromaKey value.

The following example uses rn:chromaKey to turn the hexadecimal color #808080 transparent. The rn:chromaKeyTolerance attribute specifies a 1-value tolerance both above and below the designated red value of 80. So in this case, the colors #7F8080 and #818080 are rendered transparent along with #808080:

```
<img rn:chromaKey="#808080" rn:chromaKeyTolerance="#010000"…/>
```

Setting Red, Green, and Blue Tolerances

In most cases, you'll want to specify tolerance ranges for red, green, and blue. When you do this, only the colors that fall within the overall range set by all the designated tolerances are rendered transparent. For example, the following three attribute pairs are all equivalent, but use different color values, which are described in Appendix D:

```
rn:chromaKey="rgb(128,128,128)" rn:chromaKeyTolerance="rgb(1,2,3)"
rn:chromaKey="rgb(50%,50%,50%)" rn:chromaKeyTolerance="rgb(0.4%,0.8%,1.2%)"
rn:chromaKey="#808080" rn:chromaKeyTolerance="#010203"
```

All of the preceding examples define a 1-value tolerance around the specified red value, a 2-value tolerance around the designated green value, and a 3-value tolerance around the selected blue value. Therefore, the colors that have the following RGB values are rendered transparent:

- red RGB values 127-129
  (red hexadecimal values of 7F, 80, and 81)

  –And–

- green RGB values 126-130
  (green hexadecimal values of 7E, 7F, 80, 81, and 82)

  –And–

- blue RGB values 125-131
  (blue hexadecimal values of 7D, 7E, 7F, 80, 81, 82, and 83)

So, for example, the following colors would be rendered transparent because they fall within the range specified by all three tolerance settings:

- `rgb(127,128,130)`, which is equivalent to hexadecimal `#7F8082`
- `rgb(128,129,125)`, which is equivalent to hexadecimal `#80817D`

However, the following colors would not be rendered transparent because they fall outside the overall range defined by all the red, green, and blue tolerance values:

- `rgb(126,126,126)`, which is equivalent to hexadecimal `#7E7E7E`

  This color is not rendered transparent because the red value falls outside the designated red tolerance, even though the green and blue values fall within the designated green and blue tolerances.

- `rgb(127,128,132)`, which is equivalent to hexadecimal `#7F8084`

  This color is not rendered transparent because the blue value falls outside the designated blue tolerance, even though the red and green values fall within the designated red and green tolerances.

Tips for Expanding the Color Transparency Range

- The `rn:chromaKeyTolerance` attribute is always used in conjunction with `rn:chromaKey`. If you use `rn:chromaKeyTolerance` without also specifying `rn:chromaKey`, the `rn:chromaKeyTolerance` value is ignored.
- Although you can use any type of color value described in Appendix D for `rn:chromaKeyTolerance`, RGB percentages are generally the simplest means for expanding the transparency range. Instead of precisely determining in advance the range of colors you want to render transparent, select your `rn:chromaKey` value, then widen the range with a small percentage value:

  ```
  rn:chromaKey="rgb(45,199,132)" rn:chromaKeyTolerance="rgb(5%,5%,5%)"
  ```

  Check the results by playing the SMIL file in RealONE Player, and adjust the various percentage values through trial-and-error until you achieve your desired result.

- As explained in the preceding section, a color must fall within the full range of the red, green, and blue tolerances to be rendered transparent. If you want to match all values for red, green, or blue, set its respective tolerance to the maximum. Consider the following equivalent examples:

  ```
  rn:chromaKey="rgb(128,128,128)" rn:chromaKeyTolerance="rgb(1,255,255)"
  rn:chromaKey="#808080" rn:chromaKeyTolerance="#01FFFF"
  ```

Both of these examples render transparent any color that has a red value in the RGB range of 127 to 129 (7F, 80, 81), regardless of that color's blue and green values.

Substituting a Color for Transparency

For clips that include transparency, such as GIF and PNG images, you can use `bgcolor` to substitute a color for the transparency. This attribute uses a `<param/>` tag, requiring the use of binary clip source tags. The value must be a hexadecimal color value without a leading pound sign (#), as shown in this example:

```
<img src="button.gif"...>
  <param name="bgcolor" value="BB21AA"/>
</img>
```

For More Information: For background information on binary tags, see "Binary and Unary Tags" on page 137.

PART

IV

ORGANIZING A PRESENTATION

Using SMIL, you can pull together simple or highly complex presentations. Chapter 9 shows how to make presentations accessible to all viewers. Chapter 10 explains how to group clips together to set up the basic presentation timeline. You'll also need to know how to organize the onscreen layout, as described in Chapter 11.

CHAPTER

9

PRESENTATION INFORMATION

RealONE Player provides several means for delivering information about a presentation, such as its title, author, and copyright. This chapter covers these information features, and explains the accessibility features available for sight-impaired persons.

Understanding Presentation Information

There are several types of presentation information available. Some types augment other types, some types override other types, and some types are available only to viewers who have devices that read accessibility information.

Information Encoded in Clips

Many clips have their own encoded information. When you create a RealVideo or RealAudio clip, for example, you can have RealProducer encode certain types of information into the clip. Some of this information is used only by Internet search engines, but some is read by RealONE Player. The following are the most common types of information encoded into clips for display by RealONE Player:

- title
- author
- copyright
- abstract (also called "description")

In general, it's good practice always to encode information in the clip. This ensures that important information, such as a copyright, is present if the clip is not streamed using SMIL. Encoded information is the most basic level of presentation information, but you can override it using SMIL.

For More Information: See the documentation for your production tool for instructions on how to encode information into a clip.

Clip Source Tag and Group Information

A SMIL clip source tag, such as `<video/>` or `<ref/>`, can define title, author, copyright, and abstract information for the clip. There are two main advantages to defining this information in SMIL:

- You can provide information for any clip, which is handy for clips that do not encode any information internally.
- The SMIL information overrides the encoded clip information, letting you modify information without re-encoding the clip.

You can also define title, author, copyright, and abstract information for groups. This information then overrides the information defined for the individual clips. When several clips play in parallel, for example, RealONE Player does not display the title for each clip individually. You may therefore want to define a single group title that RealONE Player displays while the group is active.

For More Information: The section "Adding Clip and Group Information" on page 168 describes how to add information to clip source tags and group tags. For more on groups, see Chapter 10.

SMIL Presentation Information

Within a SMIL file, you can define information for the entire presentation. This information supplements the clip or group information, but does not override it. This enables you to present two-levels of information to viewers:

- The presentation information lasts for the entire presentation.
- The clip or group information lasts only as long as each clip or group plays.

Like clip information, the presentation information can give the title, author, copyright, and abstract. But you can also define any other information you wish through the header section `<meta/>` tags.

For More Information: The section "Defining Information for the SMIL Presentation" on page 169 explains how to write the `<meta/>` tags.

Accessibility Information

The accessibility features define a different class of information. RealONE Player typically does not display this information. Instead, the information is read by assistive devices used by sight-impaired persons. This information can help these viewers choose which clips to play, and which links to click.

For More Information: See "Adding Accessibility Information" on page 169 for more information on these features.

RealONE Player Context Window

RealONE Player on Windows has a built-in context window meant for displaying information as a presentation plays. Through SMIL, you can open HTML pages in the context window at any point in a presentation. On operating systems other than Windows, these pages open in the viewer's default Web browser.

For More Information: See "Opening HTML Pages in the Context Window" on page 297 for more information.

Coded Characters

In a SMIL header section, or within clip attribute values, quotation marks, apostrophes, ampersands, and angle brackets are interpreted as syntax markers. You need to use codes to have these characters show up as text in RealONE Player. As shown in the following table, codes begin with an ampersand ("&") and end with a semicolon (";"). SMIL interprets these codes the same way as popular Web browsers.

SMIL Coded Characters

Code	Character	Example
`"`	quotation mark	"
`&`	ampersand	&
`'`	apostrophe	'

(Table Page 1 of 2)

SMIL Coded Characters (continued)

Code	Character	Example
<	left angle bracket ("less than" sign)	<
>	right angle bracket ("greater than" sign)	>

(Table Page 2 of 2)

For example, to add the following as a title:

```
Multimedia's <smil> & you
```

You enter this in the SMIL file:

```
"&amp;quot;Multimedia&amp;apos;s &amp;lt;smil&amp;gt; &amp;amp; you&amp;quot;"
```

Adding Clip and Group Information

The title, author, copyright, and abstract attributes let you add information to clip source tags and group tags. The following table summarizes these descriptive attributes.

Clip and Group Information Attributes

Attribute	Value	Function
abstract	*clip_abstract*	Sets summary displayed in RealONE Player.
author	*author_name*	Defines author name.
copyright	*copyright_notice*	Provides copyright notice.
title	*title_text*	Creates a title that displays in the playlist.

You can use any combination of these attributes in each group or clip source tag, but RealNetworks highly recommends that you always include title attribute values, which appear in the RealONE Player playlist. Each attribute takes a text string for its value. The following example shows the general form these attributes take in a clip source tag:

```
<ref src="..." title="title" author="name" copyright="date" abstract="abstract"/>
```

Tips for Defining Clip Information

The following points explain the relationship between clip information set through SMIL and information encoded in a clip.

- For a sequence or parallel group, use title, author, copyright, and abstract attributes in the <par> or <seq> tag instead of the clip source tags. With a

parallel group or sequence, clip titles are ignored, and only the group title shows in the RealONE Player playlist.

- A title, author, copyright, or abstract attribute in an <excl> tag is ignored. This information must be defined for individual clips or parallel groups.
- If a title is not encoded in the clip or specified through a title attribute, the clip's URL is used in the RealONE Player playlist and the "recent clips" list.

Defining Information for the SMIL Presentation

Whereas clip source tags can define information about each clip, the SMIL file header can use <meta/> tags to define information, such as title, author, and copyright, for the entire presentation. Each <meta/> tag uses two attributes, name and content, as shown in the following example:

```
<head>
  <meta name="title" content="Bob and Susan Discuss Streaming Media"/>
  <meta name="author" content="RealNetworks Media Productions"/>
  <meta name="copyright" content="(c)1998 RealNetworks"/>
  <meta name="abstract" content="Bob and Susan, two Internet technology
experts, discuss the future of streaming media."/>
 </head>
```

Tip: Name values, as in name="title", must be lowercase. When defining long content such as an abstract, don't use line breaks or tabs within a content value.

Managing Presentation Information

This section will be added in a later version of this guide.

Adding Accessibility Information

Whereas some attributes are displayed in RealONE Player, others function only with assistive reading devices used by visually impaired viewers. The following table summarizes the attributes that help make your presentation

accessible to all viewers. RealNetworks encourages you to add these attributes to your presentation.

Accessibility Attributes

Attribute	Value	Function	Reference
alt	*text*	Provides alternate text.	page 170
longdesc	*text*	Gives a long description to assistive reading devices.	page 170
readIndex	*integer*	Sets the order in which clip information is read.	page 171

Including an Alternate Clip Description

Each clip source tag can include an alt attribute that uses short, descriptive text as its value. This alt value displays in RealONE Player when the viewer moves the screen pointer over the clip. It is good practice always to include an alt attribute for each clip. In the following example, the text "Introductory Video" displays when the viewer moves the screen pointer over the clip:

```
<video src="video1.rm" alt="Introductory Video"/>
```

> **Note:** Unlike browsers that display image alt text before the images are downloaded, RealONE Player does not display alt text for clips before they play.

> **Tip:** If the clip includes hyperlinks, the link's alt value or URL displays in place of the clip's alt text. For more on alt in hyperlinks, see "Displaying Alternate Link Text" on page 293.

Using a Long Description

Each source tag can include a longdesc attribute that supplements the alt attribute. Some assistive-reading devices can read this long description for visually-impaired viewers. If you turn the clip into a hyperlink as described in Chapter 14, the description should describe the link destination. Here is an example:

```
<img src="button3.gif" longdesc="This is the third navigation button. Clicking it
opens your browser to the home page for RealNetworks." .../>
```

Setting the Clip Read Order

When a visually-impaired viewer uses an assistive-reading device, the device typically reads the values of the title, alt, and longdesc attributes in each clip source tag. When clips play in parallel, the device reads the attributes in the order that the clip tags appear in the <par> group. To change this order, you can add readIndex attributes to the clip source tag. Each readIndex attribute, which has a default value of 0, takes a positive integer as a value. Here is an example:

```
<par>
  <img ... alt="Link Button 1" longdesc="Start next video" readIndex="1"/>
  <img ... alt="Link Button 2" longdesc="Visit home page" readIndex="2"/>
  <video ... alt="Presentation Video" readIndex="0"/>
</par>
```

In the example above, the video source tag has the lowest readIndex value, so an assistive device reads that clip's alt attribute information first. Next, the device reads the first image's alt and longdesc attributes, followed by the second image's alt and longdesc attributes.

> **Note:** If two or more source tags have the same readIndex value, clip information is read according to the order that the clip source tags appear in the markup.

> **For More Information:** The section "Playing Clips in Parallel" on page 177 describes parallel groups.

CHAPTER 10

GROUPS

Grouping clips is the fundamental way to organize a presentation timeline. For example, you can play clips one after another, or display several clips at the same time. This chapter describes how to use the basic group tags to organize a presentation. Once you understand how groups work, you can use the timing commands described in later chapters to modify group behavior.

Understanding Groups

Within a SMIL presentation, you can organize clips into three types of groups. The presentation can have any number of these groups:

- sequences

  In a sequence, clips play one at a time, one after the other. When one clip stops, the next clip begins, and so on until the sequence finishes. In SMIL, a `<seq>` tag indicates the start of a sequence. A corresponding `</seq>` tag denotes the end of the sequence. The section "Playing Clips in Sequence" on page 175 explains sequences.

- parallel groups

  In a parallel group, all clips play together. For example, a parallel group could include a video and a RealText clip that provides subtitles. When you create a parallel group, you need to define a layout that specifies where each clip appears onscreen. A `<par>` tag starts a parallel group, and a `</par>` tag ends the group. See the section "Playing Clips in Parallel" on page 177 for more information.

- exclusive groups

  In an exclusive group, only one clip plays at a time. This type of group is typically created for interactive presentations. For example, a presentation may include several buttons, each of which selects a different video. Depending on which button the viewer clicks, a different clip from the

group is selected. An exclusive group is created between <excl> and </excl> tags. The section "Creating an Exclusive Group" on page 186 explains exclusive groups.

Groups Within Groups

One of the powerful features of SMIL is the ability to nest groups within groups. You can combine <seq> and <par> tags as needed to create any type of presentation, for example. The organization of these tags greatly affects the presentation playback, though, and you need to be careful when creating nested groups within a large presentation. In the following example, clip 1 plays first. When it finishes, clip 2 and clip 3 play together. When both clip 2 and clip 3 have finished playing, clip 4 plays:

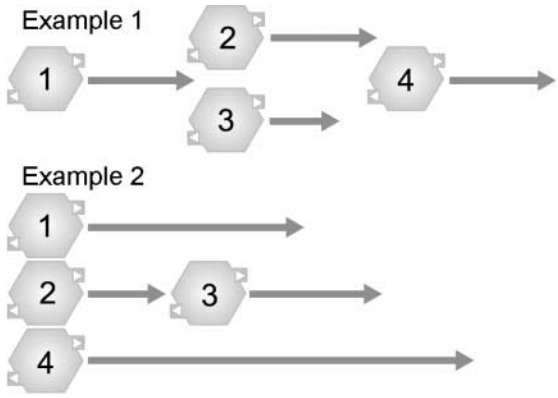
```
<seq>
   clip 1
   <par>
      clip 2
      clip 3
   </par>
   clip 4
</seq>
```

You get very different results, though, if you switch the <seq> and <par> groupings. In the next example, clips 1, 2, and 4 all begin at the same time. When clip 2 finishes, clip 3 starts:

```
<par>
   clip 1
   <seq>
      clip 2
      clip 3
   </seq>
   clip 4
</par>
```

The following illustration shows the difference between these groupings.

Different Playback Results with Nested Groups



Playing Clips in Sequence

A sequence is the simplest type of group to create. Simply list the clips in the order you want them to play within <seq> and </seq> tags. The following example shows the entire SMIL markup required to play three audio clips in sequence:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <body>
    <seq>
      <audio src="song1.rm"/>
      <audio src="song2.rm"/>
      <audio src="song3.rm"/>
    </seq>
  </body>
</smil>
```

In the preceding example, the second clip begins when the first clip finishes, and the third clip begins when the second clip finishes. A sequence can include any number of clips, and the clips can be of any type. You could add a RealAudio or Flash clip to the sequence shown above, for example. When using visual clips, however, you should also define a layout as described in Chapter 11.

When you enclose clips in <seq> and </seq> tags, RealONE Player treats the sequence as a single presentation. If each clip in the preceding example is two minutes in length, for example, the RealONE Player status bar indicates that the presentation is six minutes long. Because RealONE Player treats the sequence as a single presentation, viewers can use the timeline slider to seek

through all the clips, but cannot choose individual clips through the RealONE Player **Play>Next Clip** command.

Creating Sequences Without <seq> Tags

It is not always necessary to group clips within <seq> and </seq> tags. Whenever clips are not listed in a group, RealONE Player automatically plays them in sequence. For instance, the following markup, which has no <seq> and </seq> tags, plays three audio clips in sequence just like the preceding example:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
 <body>
  <audio src="song1.rm"/>
  <audio src="song2.rm"/>
  <audio src="song3.rm"/>
 </body>
</smil>
```

When you do not use a <seq> group, however, RealONE Player treats each clip as a separate presentation. Suppose that each clip in the preceding example lasts two minutes. When the sequence starts, the RealONE Player status bar indicates that the presentation lasts two minutes. When the first clip ends, RealONE Player's timeline slider resets, the second clip starts, and the status bar indicates another two-minute presentation. This action repeats when the third clip plays. At any point, the viewer can select a different clip with the RealONE Player **Play>Next Clip** command.

Tips for Creating Sequences

- A <seq> tag can include a title, author, copyright, or abstract attribute just like a clip source tag. For more information, see "Adding Clip and Group Information" on page 168.
- A sequence and each clip within a sequence can use a begin attribute to delay playback. For more information, see "Setting Begin and End Times" on page 240.
- A sequence and each clip within a sequence can use a dur attribute to control the total playing time. For more information, see "Setting Durations" on page 243.
- You can use the repeatDur and repeatCount attributes to repeat a sequence or a clip within a sequence. See "Repeating an Element" on page 247 for more information.

- Because clip source tags as well as the <seq> tag can have timing attributes, it is easier to set all necessary timing attributes in the clip source tags first, so that they operate as you want them to within the sequence. Then, after determining how long the sequence will last, use timing attributes within the <seq> tag to modify the group behavior if necessary.

Playing Clips in Parallel

You can play two or more clips at the same time by grouping the clip source tags between <par> and </par> tags. The following example creates a parallel group that combines a RealVideo clip with a RealText clip:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <head>
    <layout>
      ...region layout defined as described in Chapter 11...
    </layout>
  <body>
    <par>
      <video src="song.rm" region="region1_ID"/>
      <textstream src="lyrics.rt" region="region2_ID"/>
    </par>
  </body>
</smil>
```

In the preceding example, the RealVideo and the RealText clips play at the same time. A parallel group can include any number of clips, but you need to define a playback region for each visual clip as described in Chapter 11. (Audio clips do not need to play in regions.) Each region defined in the layout must have a unique id="*ID*" attribute. You then assign each clip to a region with a region="*ID*" attribute in the clip source tag.

Tips for Creating Parallel Groups

- When you create parallel groups, you need to be careful that clips playing at the same time do not exceed the audience connection's maximum bandwidth, which is described in "Audience Bandwidth Targets" on page 32. If the maximum streaming bandwidth is 34 Kbps, for example, do not have two clips that each stream 20 Kbps of data play in parallel.

- A <par> tag can include a title, author, copyright, or abstract attribute just like a clip source tag. For more information, see "Adding Clip and Group Information" on page 168.
- A parallel group normally lasts as long as the longest clip in the group. However, you can modify this with the endsync attribute, as described in "Ending a Group on a Specific Clip" on page 245.
- A parallel group and each clip within the group can use a begin attribute to delay playback. For more information, see "Setting Begin and End Times" on page 240.
- A parallel group and each clip within the group can use a dur attribute to control the total playing time. For more information, see "Setting Durations" on page 243.
- You can use the repeatDur and repeatCount attributes to repeat a parallel group. See "Repeating an Element" on page 247 for more information.
- By using readindex attributes, you can change the order that assistive reading devices read attributes of clips in parallel groups. See "Setting the Clip Read Order" on page 171 for more information.

Synchronizing Playback in Parallel Groups

Under normal circumstances, RealSystem keeps clips within a parallel group synchronized, as long as you have authored your presentation so that its timeline runs smoothly, and it doesn't consume more bandwidth than its target audience has available. The following table summarizes the optional attributes you can add to elements within parallel groups to modify playback behavior, especially under adverse conditions.

Parallel Group Synchronization Attributes

Attribute	Value	Function	Reference
syncBehavior	canSlip|default|independent|locked	Determines if clips can fall out of synchronization.	page 179
syncBehaviorDefault	canSlip|independent|inherit|locked	Sets default synchronization for a group.	page 182
syncTolerance	default|*time_value*	Loosens synchronization for locked elements.	page 185
syncToleranceDefault	default|*time_value*	Sets a default synchronization tolerance for a group.	page 185

Creating an Independent Timeline

Adding syncBehavior="independent" to a clip in a parallel group keeps the clip completely unsynchronized from other clips in the group. In fact, the clip acts like a live broadcast. Moving the RealONE Player timeline slider does not fast-forward or rewind the clip. In the following parallel group, the RealText clip has an independent synchronization behavior. It begins to play along with the RealAudio and Flash clips, but if the viewer fast-forwards or rewinds the presentation, only the RealAudio and Flash clips are affected:

```
<par>
  <audio src="soundtrack.rm" .../>
  <ref src="training.swf" .../>
  <textstream src="translation.rt" syncBehavior="independent" .../>
</par>
```

Note that a parallel group's overall timing still applies to a clip that uses syncBehavior="independent". In the following example, the parallel group plays first, lasting for five minutes because of the dur attribute in the <par> tag. A video then follows the group in sequence. If the viewer moves the timeline slider to the five-minute mark, for instance, all clips in the parallel group end, and the video plays. So even if it lasts 10 minutes, the RealText clip ends when the group ends, regardless of its syncBehavior="independent value:

```
<body>
  <seq>
    <par dur="5min">
      <audio src="soundtrack.rm" .../>
      <ref src="training.swf" .../>
      <textstream src="translation.rt" syncBehavior="independent" .../>
    </par>
    <video src="conclusion.rm" .../>
  </seq>
</body>
```

The independent value is the only syncBehavior value that has a visible effect on how a parallel group plays under normal circumstances. As described in the following sections, the other syncBehavior values affect clips in a parallel group only under difficult streaming conditions.

Setting the Synchronization Behavior

RealSystem generally compensates well for changing network conditions to keep a presentation streaming smoothly. Under highly adverse conditions,

though, it may have to suspend playback of a group until more data arrives. With the syncBehavior attribute, you can influence how RealSystem handles these situations. Think of these attributes as defensive measures: they don't affect how your presentation plays under normal circumstances, just how it handles adverse situations. The following table describes the attribute values.

syncBehavior Attribute Values

Value	Function
canSlip	RealONE Player can suspend playing this clip as long as necessary until more clip data arrives. It then fast-forwards the clip so that it catches up with the group timeline. Other clips continue playing without regard to the state of this clip.
locked	The clip must stay synchronized with the group. If the clip's data stream stops, RealONE Player halts the group playback until new clip data arrives. You can also add a tolerance value, as described in "Loosening the Synchronization for Locked Elements" on page 185.
independent	Clip playback is entirely independent of group playback. See "Creating an Independent Timeline" on page 179.
default	The clip behavior is controlled by the group tag's syncBehaviorDefault attribute, as described in "Specifying Synchronization Behavior Default Values" on page 182. You do not need to set this value explicitly if you also set a syncBehaviorDefault value because clips will inherit the default value automatically.

Note: If you do not set any syncBehavior values, elements behave as if they are set to the canSlip value.

Synchronizing Clips

In most cases, you'll want to use a combination of canSlip and locked as the syncBehavior value for clips within parallel groups. Consider the following example, in which a Flash clip, a RealAudio clip, and a RealText clip play in parallel. This example could be a training movie in which the Flash animation displays visual information, the RealAudio clip provides an audio narration, and the RealText clip supplies translated audio subtitles:

```
<par>
  <audio src="soundtrack.rm" syncBehavior="locked" .../>
  <ref src="training.swf" syncBehavior="canSlip" .../>
  <textstream src="translation.rt" syncBehavior="locked" .../>
</par>
```

The Flash clip in the preceding example can slip, meaning that RealONE Player will suspend playback for this clip first if bandwidth drops too low. RealONE Player will resume playing the clip when more bandwidth is available. At that point, it will fast-forward the Flash clip to bring it into synchronization with the RealAudio and RealText clips. The viewer will notice that the Flash clip has paused, but the audio and the subtitles will continue to play as long as conditions do not get too bad.

The RealAudio soundtrack and the RealText subtitles in the preceding example are locked with the group. This means that RealONE Player does everything it can to keep these clips synchronized and flowing smoothly. As described above, RealONE Player's first suspends the Flash clip if necessary. If that action does not provide enough bandwidth, and the RealAudio stream also runs dry, RealONE Player halts the entire group until it has received enough data to continue playing the RealAudio and the RealText clips. It brings the Flash clip up to the group's current position whenever conditions allow.

Synchronizing Groups

Just as you can synchronize clips within groups, you can also synchronize groups within groups. Suppose that the preceding example is modified so that it plays a sequence of RealAudio clips, rather than just one clip, in parallel with the Flash and RealText clips:

```
<par>
  <seq syncBehavior="locked">
    <audio src="soundtrack1.rm" .../>
    <audio src="soundtrack2.rm" .../>
    <audio src="soundtrack3.rm" .../>
  </seq>
  <ref src="training.swf" syncBehavior="canSlip" .../>
  <textstream src="translation.rt" syncBehavior="locked" .../>
</par>
```

In the preceding example, the `syncBehavior` attribute is used in the `<seq>` tag to lock the entire sequence of RealAudio clips with the parallel group. Because of group nesting, synchronization can become complex, as shown in the following abstract example:

```
<par id="master_group">
  <ref id="clip_A" syncBehavior="locked" .../>
  <par id="group_X" synchBehavior="locked" .../>
    <ref id="clip_B" synchBehavior="locked" .../>
    <ref id="clip_C" synchBehavior="canSlip" .../>
  </par>
  <par id="group_Y" syncBehavior="canSlip">
    <ref id="clip_D" synchBehavior="locked" .../>
    <ref id="clip_E" synchBehavior="locked" .../>
  </par>
</par>
```

To understand how this hypothetical grouping works, look at the outer `<par>` group first. You can see that this group contains three elements: `clip_A`, `group_X`, and `group_Y`. The `syncBehavior` attributes on these elements determine the presentation's overall synchronization. Because `clip_A` and `group_X` are locked, RealONE Player ensures that these elements stay synchronized. Under adverse conditions, it first halts playback of `group_Y` if necessary.

Within `group_X`, `clip_B` is locked. Hence, `clip_B` will continue to play in step with `clip_A` unless network conditions greatly deteriorate. Because `clip_C` can slip, RealONE Player's second line of defense is to halt playback for `clip_C` while keeping `clip_A` and `clip_B` playing. When network conditions improve, RealONE Player first restores `clip_C`, then `group_Y` to the presentation. Note, however that both `clip_D` and `clip_E` are locked in `group_Y`. This means that RealONE Player won't restore `group_Y` until it can play both clips.

Specifying Synchronization Behavior Default Values

The group attribute `syncBehaviorDefault` is useful for setting synchronization behaviors with groups that contain many clips, or with nested groups. In a group tag, the `syncBehaviorDefault` attribute determines which synchronization behavior is used if a group element does not specify a `syncBehavior` value, or uses `syncBehavior="default"`. In the following example, the group tag has a `locked` synchronization behavior set by default:

```
<par syncBehaviorDefault="locked">
  <audio src="soundtrack.rm"  .../>
  <ref src="training.swf" syncBehavior="canSlip" .../>
  <textstream src="translation.rt" syncBehavior="default" .../>
</par>
```

In the preceding example, the RealAudio clip does not specify a `syncBehavior` attribute, and the RealText clip uses `syncBehavior="default"`. Both clips

therefore use the default value (locked) set in the group tag. The Flash clip specifies a different synchronization behavior, though, which overrides the default setting.

Setting Groups to Inherit Synchronization Defaults

A group tag's syncBehaviorDefault attribute can have the values locked, canSlip, or independent, which are described in the table "syncBehavior Attribute Values" on page 180. The attribute's default value is inherit, which you can also set explicitly in a group tag. This inherit value is useful with nested groups, as shown in the following abstract example:

```
<par id="master_group" syncBehaviorDefault="canSlip">
  <par id="group_X" syncBehaviorDefault="inherit">
    ...group_X clips played in parallel...
  </par>
  <par id="group_Y">
    ...group_Y clips played in parallel...
  </par>
  <par id="group_Z" syncBehaviorDefault="locked">
    ...group_Z clips played in parallel...
  </par>
</par>
```

In this example, group_X and group_Y both inherit the master group's syncBehaviorDefault value of canSlip. Elements within these two groups will use the canSlip behavior unless another value is specified in their tags. On the other hand, group_Z overrides the master group's behavior and sets a default of locked. Elements within group_Z will use the locked behavior unless they explicitly specify a different value.

Nested Group Interactions with Synchronization Behaviors

When you have several levels of nested groups that use syncBehavior and syncBehaviorDefault, it's important to understand how the groups and their elements interact. Because elements inherit a syncBehaviorDefault value by default, the interactions can be difficult to grasp unless you look at all levels of the nested groups. Consider the following abstract example:

```
<par id="master_group" syncBehaviorDefault="canSlip">
  <par id="group_X" syncBehaviorDefault="inherit">
    <ref id="clip_A" .../>
    <ref id="clip_B" syncBehavior="locked" .../>
  </par>
  <par id="group_Y" syncBehavior="locked">
    <ref id="clip_C" .../>
    <ref id="clip_D" .../>
  </par>
  <par id="group_Z" syncBehaviorDefault="locked">
    <ref id="clip_E" .../>
    <ref id="clip_F" syncBehavior="locked" .../>
  </par>
</par>
```

The master group sets a syncBehaviorDefault value of canSlip. The elements within this master group have the following syncBehavior values:

- group_X set to canSlip

  group_X inherits the default value of canSlip from master_group, and passes that value to the clips it contains, one of which overrides the value:

  - clip_A set to canSlip
  - clip_B set to locked

- group_Y set to locked

  group_Y sets its own behavior to locked. However, it inherits the default value of canSlip from master_group, and passes that value to both clips it contains:

  - clip_C set to canSlip
  - clip_D set to canSlip

- group_Z set to canSlip

  group_Z inherits the default value of canSlip from master_group. However, it changes the default value for the elements it contains to locked. One of the clips overrides that value:

  - clip_E set to locked
  - clip_F set to canSlip

Loosening the Synchronization for Locked Elements

When you use syncBehavior="locked" on elements within a group, RealSystem keeps those elements, whether clips or other groups, tightly synchronized. You can loosen the synchronization by adding syncTolerance="*time_value*" to the containing group. A tolerance value is useful if the elements do not need to be highly synchronized: the higher the tolerance, the less likely that RealONE Player will have to halt the entire group to rebuffer data. The following example adds a three-second tolerance to the locked elements:

```
<par syncTolerance="3s">
  <audio src="soundtrack.rm" syncBehavior="locked" .../>
  <ref src="training.swf" syncBehavior="canSlip" .../>
  <textstream src="translation.rt" syncBehavior="locked" .../>
</par>
```

In the preceding example, the locked RealAudio and RealText clips can fall at least three seconds out of synchronization before RealONE Player stops the group to rebuffer the data streams. Base the amount of time to set for a tolerance on your judgment of how far the clips can fall out of synchronization without the group playback becoming too confusing for the viewer.

> **For More Information:** SMIL timing values are described in "Specifying Time Values" on page 239.

Specifying Synchronization Tolerance Default Values

Similar to syncBehaviorDefault, the group attribute syncToleranceDefault can set synchronization tolerances for nested groups. In the following example, the master containing group sets a syncToleranceDefault value of three seconds:

```
<par id="master_group" syncTolerance="4s" syncToleranceDefault="3s">
  <par id="group_X" syncBehavior="locked" syncTolerance="inherit">
    ...group_X clips played in parallel...
  </par>
  <par id="group_Y" syncBehavior="canSlip">
    ...group_Y clips played in parallel...
  </par>
  <par id="group_Z" syncBehavior="canSlip" syncTolerance="5s">
    ...group_Y clips played in parallel...
  </par>
</par>
```

In the preceding example, group_X includes syncTolerance="inherit" and group_Y does not have a syncTolerance value. Both groups therefore inherit the master group's tolerance value of three seconds. However, group_Z sets its own tolerance value of five seconds, which overrides the master group's default.

Note that the master group has both a syncTolerance and a syncToleranceDefault value. When you use synchronization tolerance values, it's important to keep in mind what these values do:

- The syncTolerance value determines the tolerance value used for elements within the group. In the preceding example, the syncTolerance value for the master group affects the tolerance applied to group_X, group_Y, and group_Z, but not to the elements within those groups.
- The master group's syncToleranceDefault value sets the tolerance on each subgroup's elements, as long as group_X, group_Y , or group_Z inherits the value and does not override it with its own tolerance value.

Tips for Synchronizing Clips

- Authoring a presentation so that it does not consume too much bandwidth is the best defense against network uncertainties. Make sure that you understand timeline and bandwidth issues as described in "Chapter 2: Presentation Planning" beginning on page 23.
- Within a parallel group, it's best to use a locked synchronization on the clip that provides the audio. Viewers are more likely to stay tuned to a presentation in which visuals stop and start if the audio continues to flow smoothly.
- If you use a locked synchronization on all clips in a group, it's a good idea to set a tolerance of a few seconds. This helps RealONE Player to prevent the entire presentation from halting if data for just one clip is slow to arrive.

Creating an Exclusive Group

The <excl> and </excl> tags create an exclusive group in which only one element plays at a time. In a <seq> group, only one element plays at a time, too, but the playback order always proceeds from the first to the last element. In contrast, an <excl> group has no predefined playback order. The playback

order depends wholly on the SMIL timing commands defined for each element in the group.

You use an exclusive group for different purposes than you use a parallel group or a sequence. With <par> and <seq> tags, you can construct a single timeline that flows continuously throughout the entire presentation. Using an exclusive group, though, you can break up a timeline through two features: interruption and interactivity.

As an example of both interruption and interactivity, imagine a group of videos in which each video plays only when the viewer clicks an icon for the video. This is interactivity. Then, as it plays, a selected video pauses intermittently as advertising clips play, automatically resuming when each ad clip finishes. This is interruption. An exclusive group may define just one of these features, or both.

> **Tip:** To understand how exclusive groups work, you'll need to know about timing attributes. You may therefore want to read Chapter 12 and Chapter 13 first.

Defining Interactive Begin Times

Adding interactivity to a presentation is a main function of an exclusive group. In the following example, an exclusive group of videos plays in parallel with three images. All the video clips in the exclusive group use interactive begin values to start playback only when the viewer clicks an image. Hence all three images appear as soon as the parallel group becomes active, but each video does not become visible until an image is clicked:

```
<par>
  <img src="number1.gif" id="button1" .../>
  <img src="number2.gif" id="button2" .../>
  <img src="number3.gif" id="button3" .../>
  <excl dur="indefinite">
    <video src="video1.rm" begin="button1.activateEvent" .../>
    <video src="video2.rm" begin="button2.activateEvent" .../>
    <video src="video3.rm" begin="button3.activateEvent" .../>
  </excl>
</par>
```

Note that the exclusive group in the preceding example uses dur="indefinite", which keeps the group active indefinitely. This is required because an <excl> has an intrinsic duration of 0 seconds when its elements use interactive

timing. You therefore need to use timing commands in the <excl> tag to control the group's overall duration.

> **For More Information:** The begin value used to start a clip with a mouse click is described in "Defining a Mouse Event" on page 269.

Using Clip Interruption

The following example demonstrates a simple exclusive group with basic clip interruption. As with a <seq> group, only one clip from this <excl> group will play at a time. Unlike a <seq> group, though, the order in which you list the clips does not matter because the timing attributes completely control playback. In the following example, clips play in the reverse order from which they are listed:

```
<excl>
  <img src="number1.png" region="images" begin="6s" dur="5s"/>
  <img src="number2.png" region="images" begin="3s" dur="5s"/>
  <img src="number3.png" region="images" begin="0s" dur="5s"/>
</excl>
```

In the preceding example, number3.png plays first. Its begin="0s" value means that it plays as soon as the <excl> group becomes active. Its dur="5s" value makes it play for five seconds. The number2.png clip starts playing three seconds after the group becomes active, however. Because only one group element can play at a time, the begin="3s" value for number2.png overrides the dur="5s" value for number3.png. When number2.png starts to play, it stops number3.png. Likewise, when number1.png starts, it stops number2.png.

> **For More Information:** For more on the begin attribute, see "Setting Begin and End Times" on page 240. Durations are explained in "Setting Durations" on page 243.

Modifying Clip Interruption Behavior

By defining priority classes, you can control how clips in an exclusive group interrupt each other. In an exclusive group that does not use priority classes, an interrupting clip stops the interrupted clip. By defining priority classes, though, you can pause the interrupted clip instead, so that its playback resumes once the interrupting clip finishes. You define a priority class with <priorityClass> and </priorityClass> tags. Between these tags, you list the media clips within that priority class, as shown here:

```
<excl>
  <priorityClass...>
    ...clips in the higher priority class...
  </priorityClass>
  <priorityClass...>
    ...clips in the lower priority class...
  </priorityClass>
</excl>
```

When you create priority classes, the order of clips within the <excl> group becomes important. The first priority class has the highest priority, the last class has the lowest priority. All clips within a priority class have the same priority, and are called *peers*.

Once you define priority classes, you can use the attributes summarized in the following table to set the interruption behavior for clips in each class. A <priorityClass> tag can have an id attribute and any of the following attributes, but no others. You cannot include timing attributes in a <priorityClass> tag, for example.

<priorityClass> Attributes

Attribute	Value	Default	Function	Reference
peers	defer|never|pause|stop	stop	Controls how clips within the same class interrupt each other.	page 189
higher	pause|stop	pause	Determines how clips with higher priority interrupt clips in the class.	page 190
lower	defer|never	defer	Specifies how interrupting clips with lower priority affect playback.	page 191
pauseDisplay	disable|hide|show	show	Sets a clip's appearance if the clip is paused.	page 192

Controlling How Peers Interact

The peers attribute for a priority class determines how clips within that priority class interrupt each other. The peers attribute can have one of the values given in the following table.

peers Attribute Values

Value	Function
defer	An interrupting clip does not start until the currently playing clip stops.
never	An interrupting clip does not start at all.

(Table Page 1 of 2)

peers Attribute Values (continued)

Value	Function
pause	The interrupting clip pauses the playing clip. After the interrupting clip finishes, the paused clip resumes playback. The pauseDisplay attribute sets the appearance of the paused clip.
stop	The interrupting clip stops the playing clip. This is the default if you leave the peers attribute out of the <priorityClass> tag, or you do not define any priority classes within an <excl> group.

(Table Page 2 of 2)

For example, to have clips within an exclusive group pause each other instead of stop each other during interruptions, you can define a single priority class and use peers="pause" as shown here:

```
<excl>
  <priorityClass peers="pause">
    <video src="video1.rm" .../>
    <video src="video2.rm" .../>
    <video src="video3.rm" .../>
  </priorityClass>
</excl>
```

For More Information: For more on pauseDisplay, see "Specifying How Paused Clips Display" on page 192.

Setting Interactions with Higher Priority Classes

For priority classes other than the highest priority class, you can use the higher attribute in the <priorityClass> tag to determine how any clip in a higher priority class interrupts a clip in the current priority class. The higher attribute can take one of the values listed in the following table.

higher Attribute Values

Value	Function
pause	An interrupting clip from a higher priority class pauses the playing clip. After the interrupting clip finishes, the paused clip resumes playback. This is the default if you do not use the higher attribute. The pauseDisplay attribute sets the appearance of the paused clip.
stop	An interrupting clip from a higher priority class stops the playing clip.

In the following example, the first priority class (class1) has higher priority. The second priority class (class2) uses higher="stop" to specify that if a clip from class1 interrupts a clip from class2, the class2 clip will stop. Note, though,

that class2 also uses peers="pause". This means that if a clip from class2 interrupts another clip from that class, the interrupted clip will pause, not stop:

```
<excl>
  <priorityClass id="class1">
    ...clips in the higher priority class...
  </priorityClass>
  <priorityClass id="class2" higher="stop" peers="pause">
    ...clips in the lower priority class...
  </priorityClass>
</excl>
```

For More Information: For more on pauseDisplay, see "Specifying How Paused Clips Display" on page 192.

Setting Interactions with Lower Priority Classes

For priority classes other than the lowest priority class, you can use the lower attribute in the <priorityClass> tag to determine how a clip from a lower priority class acts if it attempts to interrupt a clip in the current priority class. The lower attribute can take one of the values listed in the following table.

lower Attribute Values

Value	Function
defer	An interrupting clip from a lower priority class does not start until the end of the current clip, as well as any higher-priority clips that play after the current clip. This is the default if you do not use the lower attribute.
never	An interrupting clip from a lower priority class does not play at all.

In the following example, the first priority class (class1) has higher priority and uses lower="never" to specify that if a clip from class2 attempts to interrupt a clip from class1, the class2 clip will not play at all. Note, though, that class2 also uses peers="defer". This means that if a clip from class2 interrupts another clip from that class, the interrupting clip will play after the interrupted clip finishes:

```
<excl>
  <priorityClass id="class1" lower="never">
    ...clips in the higher priority class...
  </priorityClass>
```

```
  <priorityClass id="class2" peers="defer">
    ...clips in the lower priority class...
  </priorityClass>
</excl>
```

Specifying How Paused Clips Display

When you set peers="pause" or higher="pause" in a <priorityClass> tag, you can also set the pauseDisplay attribute, which determines how a clip appears when it pauses. This attribute, which has no effect on audio-only clips, can take one of the values listed in the following table.

pauseDisplay Attribute Values

Value	Function
disable	The paused clip appears visible but disabled in RealONE Player. It does not respond to mouse clicks until it resumes playback.
hide	The paused clip disappears until it resumes playback.
show	The paused clip remains visible in RealONE Player, and it continues to respond to mouse clicks. This is the default if you do not use the pauseDisplay attribute.

In the following example, each clip that interrupts another clip causes that clip to pause and disappear. After the interrupting clip finishes playing, the interrupted clip reappears and resumes playback:

```
<excl>
  <priorityClass peers="pause" pauseDisplay="hide">
    <video src="video1.rm" .../>
    <video src="video2.rm" .../>
    <video src="video3.rm" .../>
  </priorityClass>
</excl>
```

Tips for Defining Exclusive Groups and Priority Classes

- An <excl> group can have one or several priority classes.
- Priority classes affect only interruption behavior. They have nothing to do with timing. A clip in a lower priority class can play before a clip in a higher priority class, or vice versa.

- When you use priority classes, every element in the <excl> group must belong to a priority class. You cannot mix clips that are within priority classes and clips that are outside of priority classes.
- Priority classes cannot be nested. That is, one priority class cannot contain another priority class.
- A priority class can contain clips or groups of clips.

CHAPTER

11

LAYOUT

When you stream more than one clip, you use SMIL layout tags to lay out the presentation. The layout defines where each clip appears in RealONE Player. Clips might appear side by side, for example, or stacked on top of each other. You can even play clips in windows that pop up from the RealONE Player main media window.

> **Note:** For instructions on laying out a presentation in a Web page instead of in RealONE Player, see Chapter 19.

Understanding Layouts

If your presentation is audio-only, or it displays just one clip, you do not need to create a layout. However, if you want to play successive clips in the same area, or if your presentation displays multiple clips together, you need to define a layout. The following sections provide an overview of the main layout features.

Root-Layout Area

You define a presentation's layout in a SMIL file's header section. You first create one (and only one) *root-layout* area, which sets the size of the main media window when the presentation starts. This size stays constant throughout the presentation unless the viewer manually resizes the window, or you change the root-layout size using a SMIL animation. You cannot play any clips in the root-layout area, but you can set its color.

> **For More Information:** The section "Defining the Main Media Window" on page 203 explains how to set up the root-layout area.

Playback Regions

Each clip plays in a rectangular *region*. Within the main media window, all regions lay within the root-layout area. You might define just one region that's the same size as the root-layout area, or you might set up multiple regions. Although similar to HTML frames, SMIL regions can overlap, letting you play a clip in one region on top of a background image in another region, for example.

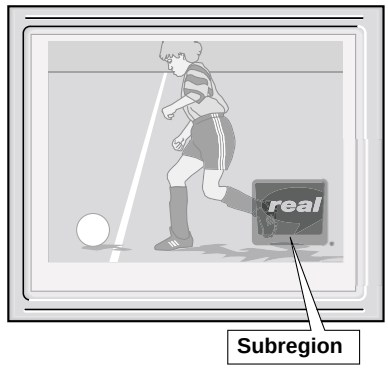
SMIL Region Possibilities

For More Information: See "Defining Playback Regions" on page 206 for information about setting up regions.

Subregions

Within each region you can also create *subregions*, which fall within their containing region, just as a region within the main media window falls within the root-layout area. A subregion automatically moves if its containing region's position changes. Using a subregion, for example, you can mimic a television channel in which a small, partially transparent channel logo appears in a corner, hovering above the content.

A Subregion Within a Region

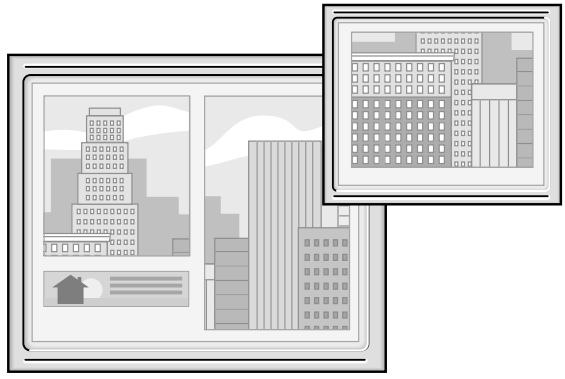


For More Information: See "Defining Subregions" on page 218 for more information on subregions.

Secondary Media Windows

Popping up above the main media window, a *secondary media window* can be moved, resized, and closed independently. You can use a secondary media window to display RealText credits for a video playing in the main media window, for example. As with the main media window, you can divide a secondary media window into separate playback regions. A secondary media window can open when the presentation starts, or when a certain clip starts to play. All clips playing in the main media window and the secondary media windows are part of the same timeline defined within a single SMIL file.

Secondary Pop-Up Window



Secondary Pop-up Windows Versus Hyperlinked Pop-up Windows

To open a new window based on viewer input, you create hypertext links to other SMIL files in your presentation. When the viewer clicks a hypertext link, RealONE Player launches a new, linked window (rather than a secondary media window) that plays a new SMIL presentation and either stops or pauses the clips in the main media window. The following table describes the differences between using a secondary pop-up window and a hyperlinked pop-up window.

Secondary Pop-up Windows Versus Hyperlinked Pop-up Windows

	Secondary Pop-up Window	Hyperlinked Pop-up Window
When does the window pop up?	The Window pops up at the beginning of the presentation or when the first clip assigned to the window begins to play.	The window pops up when the viewer clicks a hyperlink in the SMIL presentation.
Do clips in the main media window continue to play?	All clips continue to play in the main media window and the pop-up window according to the SMIL timeline.	You can choose whether to continue, pause, or stop the presentation in the main media window.
How many SMIL files do I write?	You write just one SMIL file that controls the timeline for the main media window and all secondary media windows.	You write separate SMIL files for the main media window and each hyperlinked pop-up window.
What user controls does the pop-up window have?	The pop-up window has buttons to minimize, maximize and close the window. All timeline and menu controls are on the main media window.	The pop-up window gives the viewer many playback controls and menus.
Can the pop-up window launch another pop-up window?	No, a secondary media window cannot launch another secondary media window. The main media window can launch any number of secondary media windows, though.	Yes, a hyperlinked pop-up window runs a new SMIL presentation that can launch new windows.

For More Information: Chapter 14 explains hyperlinking.

Clip Position and Fit

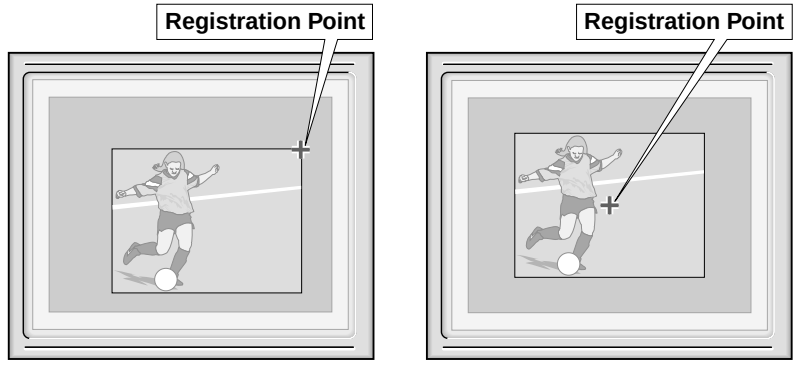
By default, a clip aligns with a region's upper-left corner and displays at its normal size. If it's too big for the region, it's cropped. If it's too small, the

region's background color displays in the remainder of the region. You can modify this behavior to align a clip to different points within a region (*clip position*), as well as resize the clip to make it fit the region (*clip fit*) better.

Clip Position

To define clip position, you create various *registration points* that specify where and how clips align to regions. One registration point might center clips in their regions, for example. Another point might align clips with their regions' bottom-left corners. The following figure illustrates two registration points, showing a few of the many ways to align clips to a point.

Registration Point Alignment



For More Information: The section "Creating Registration Points" on page 221 explains how to specify clip positions.

Clip Fit

The clip fit determines what happens when a clip is larger or smaller than its region. When a clip does not fit a region, you can keep the clip its normal size, scale the clip larger or smaller, or even add scroll bars to handle large clips.

For More Information: The section "Defining How Clips Fit Regions" on page 227 explains how to control clip fit.

Tips for Laying Out Presentations

SMIL provides many options for laying out presentations. In many cases, you can achieve the same visual layout by different methods, but some methods may provide more clip placement options, for example, or create a layout that's easier to modify. Before you lay out a presentation, make sure you

understand the options available to you. The following sections will help you make choices based on the type of presentation you want to create.

> **Tip:** It may help to sketch the layout on paper or with illustration software. In your sketch, position the regions, subregions, and clips, noting their sizes and the thickness of any borders that should appear around them.

How big should I make the root-layout area?

Unless the viewer manually resizes the main media window, it stays at the root-layout size for the duration of the presentation. Therefore, you need to make sure that the root-layout area is large enough to encompass all clips you plan to play. Calculate the root-layout size based on the sizes of clips that play together, as well as any borders you want to add.

Root-Layout Example

Suppose you plan to display two clips, one 100 pixels wide and the other 200 pixels wide, side-by-side. If you want a five-pixel border around the clips, for example, the root-layout area needs to be 315 pixels wide:

- 5 pixels from the left edge of the root-layout area to the first clip.
- 100 pixels for the first clip.
- 5 pixels from the right edge of the first clip to the left edge of the second clip.
- 200 pixels for the second clip.
- 5 pixels from the right edge of the second clip to the right edge of the root-layout area.

RealONE Player Menus and Controls

When choosing a root-layout size, keep in mind that the RealONE Player menus and controls will appear around the main media window. If you define a very large root-layout area, some parts of the main media window, or some RealONE Player controls, may not appear on the viewer's screen. The smallest computer screen is general use is 640 pixels wide by 480 pixels high.

Double-Screen and Full-Screen Modes

As described in "Setting a Presentation's Starting Mode" on page 422, you can make the presentation display at double-size or full-screen mode when it starts up. Doing this may affect how you define the root-layout area. For

example, most computer screens have a width-to-height ratio of 4:3. Therefore, a root-layout area that also has a 4:3 ratio will scale best in full-screen mode.

Should my presentation use secondary media windows?

A secondary pop-up window is a useful way to provide additional information in a presentation. You might use the window to provide hypertext links to other streaming presentations or Web pages, for example. A secondary media window also provides a way to work additional clips into your presentation without making the root-layout area too big.

Add secondary media windows with caution, though. Using too many secondary media windows may make the presentation cluttered and difficult for the viewer to follow. Keep in mind, too, that the viewer can close secondary media windows at any time. Once closed, these windows do not open again unless another clip is scheduled to play in them later, or the viewer replays the presentation. For this reason, you may not want to play crucial clips in secondary media windows.

> **Tip:** As with the root-layout area, calculate a secondary media window's height and width based on the sizes of clips that play together in the window, as well as any borders you want to add.

How many regions should I create?

Every visual clip must be assigned to a region, but you don't necessarily have to create a separate region for each clip. When you play a sequence of clips, for example, you can assign each new clip to the same region, using registration points if necessary to align each clip to the region. When multiple clips play in parallel, though, RealNetworks recommends that you define a separate region (either a main region or a subregion) for each clip.

Should I define subregions?

Any layout that uses subregions can be duplicated using just main regions. But using subregions simplifies certain layout tasks because subregions are associated with their containing regions. For example, if you move a region 10 pixels to the left in the root-layout area, all of its subregions automatically move with it. If the subregions were main regions instead, you'd have to change their layout attributes individually to keep them at the same relative position within the larger region.

Should I create registration points?

If your regions and subregions are the same sizes as the clips that play in them, you do not need to create registration points. You may want to create registration points if regions are larger than clips, however and you don't want clips to align with the regions' upper-left corners.

Can I use subregions instead of registration points to position clips?

Yes. Suppose you want to position a small clip somewhere within a large region. You could either apply a registration point to the region, or you could create a subregion inside the region.

When to Use a Registration Point

The primary advantage of defining a registration point is that you can easily apply the point to multiple regions. To center several clips in several different regions, it's much easier to define a single registration point and apply it to the various regions than to create a subregion for each clip.

When to Use a Subregion

Defining a subregion for a smaller clip gives you more options in determining how the clip appears within the region. You can set a specific subregion size, for example, and specify how the clip scales within the subregion. If you want multiple clips to overlap, you should use subregions because you can set the clips' stacking order by using the subregions' `z-index` attributes.

Layout Tag Summary

The following SMIL sample illustrates the functions and relationships of the main layout tags. Layout markup goes in the SMIL header section, between `<layout>` and `</layout>` tags:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
 <head>
  <layout>
   <root-layout ...defines the main media window's overall size.../>
   <region id="ID1" ...defines a playback region within the main window.../>
   <region id="ID2" ...defines a playback region that has a subregion...>
    <region id="ID3" ...defines a subregion.../>
   </region>
   <topLayout ...defines a secondary media window's overall size...>
    <region id="ID4" ...defines a region within the secondary window.../>
   </topLayout>
   <regPoint id="ID5" ...defines a point where clips are placed in regions.../>
```

```
  </layout>
 </head>
 <body>
  ...clips and groups...
  <ref src="..." region="ID1" regPoint="ID5" ...assigns a clip to a region
   and a registration point by IDs.../>
  <ref src="..." region="ID2" ...assigns a clip to a region by ID.../>
  ..more clips and groups...
 </body>
</smil>
```

For More Information: For more on the SMIL header and body sections, see "Header and Body Sections" on page 134.

Creating Main and Secondary Media Windows

The simplest layout defines a size for the RealONE Player main media window, and creates a single playback region for clips. More complex layouts can create multiple regions, and even launch secondary, pop-up windows. The following sections explain how to define and set the sizes for the main media window, as well as any secondary media windows you want to use.

Defining the Main Media Window

For every SMIL presentation that uses a layout, you use the `<root-layout/>` tag to set the main media window's width and height in pixels. The `<root-layout/>` tag requires `height` and `width` attributes. An `id="ID"` attribute is optional, and is generally required only if you use SMIL animations to change the window size as the presentation plays. The following example creates a root-layout area 320 pixels wide by 240 pixels high:

```
<layout>
 <root-layout width="320" height="240"/>
 ...main media window regions defined after the root-layout area...
</layout>
```

Because clips cannot play in the root-layout area, you need to define at least one region in addition to the root-layout area. In the following example, the single region automatically assumes the same size as the root-layout area:

```
<layout>
 <root-layout width="320" height="240"/>
 <region id="video_region"/>
</layout>
```

For More Information: "Defining Playback Regions" on page 206 explains how to set region sizes and positions within the main media window. "Adding Background Colors" on page 215 tells how to set window colors.

Creating Secondary Media Windows

To add secondary, pop-up windows to a presentation, you include <topLayout> and </topLayout> tags for each window you want to launch. As with the <root-layout/> tag, you specify the width and height of each secondary media window in pixels. An id="*ID*" attribute is optional, and is generally required only for use with SMIL animations. The following example creates a secondary media window 180 pixels wide by 120 pixels high, and defines a single playback region of the same size:

```
<layout>
  <root-layout.../>
  ...main media window regions defined...
  <topLayout width="180" height="120">
    <region id="popup_region"/>
  </topLayout>
</layout>
```

For More Information: See "Defining Playback Regions" on page 206 for information on setting region sizes and positions. "Adding Background Colors" on page 215 explains how to define a window color.

Controlling When Secondary Media Windows Open and Close

A <topLayout> tag can include open and close attributes that determine when the secondary media window appears and disappears. The following table describes the values these attributes can have. You can define one open value,

and one close value, or leave these attributes out of the tag to use the default values.

Attributes for Opening and Closing Secondary Media Windows

Attribute and Value	Function
open="onStart"	Open the window when the presentation begins, regardless of when clips play in the window. Keep the window open until the presentation ends or the viewer closes the window. This is the default.
open="whenActive"	Open the window when a clip begins to play in a region within the window.
close="onRequest"	Close the window only when the viewer clicks the close button. This is the default.
close="whenNotActive"	Close the window when clips stop playing in the window, or when the viewer clicks the close button.

With the default values of open="onStart" and close="onRequest", the secondary media window opens when the presentation begins (even if no clips play in the window immediately), and stays open until the viewer closes the window or starts another presentation. A common alternative is to make the window appear only when clips play in it, and close when those clips finish playing:

```
<topLayout ... open="whenActive" close="whenNotActive">
```

Tips for Defining Secondary Media Windows

- A viewer can always close a secondary media window manually, regardless of the close attribute's value. If a clip is assigned to play in a secondary media window the viewer has closed, RealONE Player still processes the streaming clip, but it doesn't display the clip's visual content. It will play any audio content, however.
- You cannot control where a secondary media window pops up on the viewer's screen. RealONE Player determines a placement based on the size of the main and the secondary media windows, as well as the arrangement of existing windows on the screen.
- If the secondary media window uses close="whenNotActive", a clip's fill attribute can affect when the window closes. For more information, see "Setting a Fill" on page 251.

- The opening or closing of a secondary media window can start or stop another element. For more information, see "Defining a Secondary Window Event" on page 273.

Changing Resize Behavior

Viewers can resize the main media window and secondary media windows manually, or by choosing RealONE Player's double-size or full-size mode. By default, all regions and clips resize accordingly. You can change this behavior, though, to allow only regions (and hence the clips within those regions) defined with percentage values to resize. In this case, clips playing in regions defined with pixel values will not resize. To do this, add the attribute `rn:resizeBehavior="percentOnly"` to the `<root-layout>` or `<topLayout>` tag:

```
<root-layout width="250" height="230" rn:resizeBehavior="percentOnly"/>
```

Using this attribute requires that you declare the following namespace in the `<smil>` tag:

```
xmlns:rn="http://features.real.com/2001/SMIL20/Extensions"
```

> **For More Information:** "Defining Region Sizes and Positions" on page 208 explains pixel and percentage values for regions. For background on customized attributes and namespaces, see "Using Customized SMIL Attributes" on page 139.

Defining Playback Regions

For the RealONE Player main media window and each secondary media window, you need to define at least one region where clips play. For the main media window, you define regions after the `<root-layout/>` tag. For secondary media windows, you define them between each window's `<topLayout>` and `</topLayout>` tags. You create each region using a `<region/>` tag:

```
<layout>
  <root-layout.../>
  <region id="ID1" ...defines a playback region within the root-layout area.../>
  <region id="ID2" ...defines a playback region within the root-layout area.../>
  <topLayout...>
    <region id="ID3" ...defines a region within a secondary media window.../>
    <region id="ID4" ...defines a region within a secondary media window.../>
  </topLayout>
</layout>
```

A <region/> tag requires only a unique ID to create a region that expands to the same size as the main or secondary media window. In most cases, though, you'll want to create smaller regions and position them within the window using other <region/> tag attributes, which are described in the following sections.

Setting Region IDs and Names

Every region must have a unique, user-defined ID in the form id="*ID*". You assign clips to a region based on the region's ID. The following SMIL example defines a region that uses video_region as its ID:

```
<layout>
  <root-layout width="250" height="230"/>
  <region id="video_region"/>
</layout>
```

Optionally, a region can include a name in the form regionName="*name*". Unlike IDs, region names do not have to be unique. In fact, region names are useful primarily when two or more regions share the same name. In this case, you can assign the same clip to play in multiple regions by using the region name rather than the region ID. The following is an example of a video clip with a region name:

```
<region id="video_region1" regionName="videoregion"/>
```

Although region names can be identical, no region name should be the same as a region ID. The following example is allowed because the IDs are unique even though the region names are identical:

```
<region id="video_region1" regionName="videoregion"/>
<region id="video_region2" regionName="videoregion"/>
```

However, the following example is not allowed because the ID is not unique:

```
<region id="video_region1" regionName="video_region1"/>
```

> **For More Information:** See "Playing the Same Clip in Multiple Regions" on page 233 and "Example 3: Resizing the Media Window for Captions" on page 381 for examples of assigning clips to regions based on the region name rather than the ID. The section "SMIL Tag ID Values" on page 138 contains rules for specifying IDs.

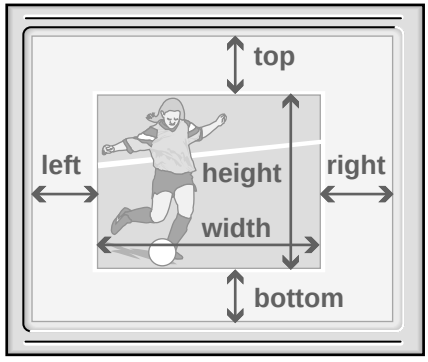
Defining Region Sizes and Positions

If you do not specify a region's size, the region becomes the same size as the window that contains it. For example, the following region expands to 320 pixels by 240 pixels, the same size as the main media window:

```
<layout>
  <root-layout width="320" height="240"/>
  <region id="video_region"/>
</layout>
```

In most cases, though, you'll want regions to be smaller than the window that contains them. This lets you place regions side-by-side, or use the window background color as a border around a region. The following figure illustrates how a region's size and position attributes control where the region appears within its window.

Region Size and Position Attributes



The region size and position attributes constitute a simple coordinate system measured in pixels or percentages. Because each attribute has a default value of auto, you can leave it out of the <region/> tag to set its value automatically based on the values of the other attributes. The result is that, in most cases, you need to specify just one to four of these attributes, rather than all six attributes listed in the following table.

Region Size and Position Attributes

Attribute	Function	Example
bottom	Sets region's bottom offset from window's bottom border.	bottom="22"
height	Specifies the region's height.	height="180"

(Table Page 1 of 2)

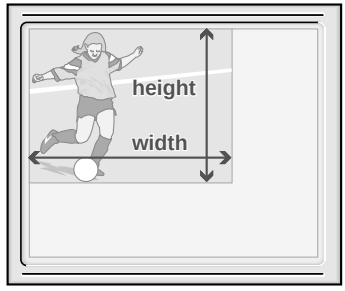
Region Size and Position Attributes (continued)

Attribute	Function	Example
left	Sets region's left offset from window's left border.	left="20%"
right	Sets region's right offset from window's right border.	right="5%"
top	Sets region's top offset from window's top border.	top="60"
width	Specifies the region's width.	width="240"

(Table Page 2 of 2)

Note: For size and position attributes, SMIL supports the use of px to designate pixels, as in top="60px". This provides consistency with the Cascading Style Sheet 2 (CSS2) standard. In SMIL, though, the px designation is not necessary. For simplicity, this manual omits the px from pixel measurements.

Layout Example 1: Region Width and Height



This example shows a region in which only the width and height are defined:

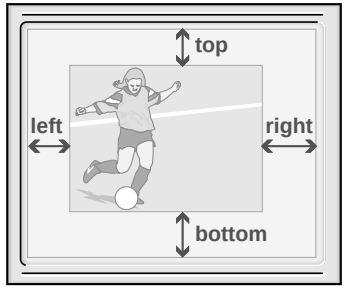
```
<region id="video_region" width="180" height="120"/>
```

In this case, the region is placed in the window's upper-left corner. The bottom and right offsets from the window borders are set automatically based on the region's size and position. If the window were 300 pixels wide by 200 pixels high, you could achieve the same layout using percentage values:

```
<region id="video_region" width="60%" height="60%"/>
```

Tip: With these percentage values, the region will change size if you modify the size of the <root-layout/> or <topLayout> tag that contains the region. With pixel measurements, though, the region size remains stable.

Layout Example 2: Four Region Offsets



This example shows a region placed in a window without specifying the region size:

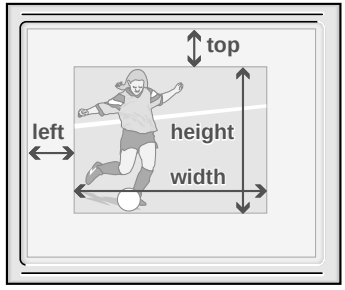
```
<region id="video_region" left="60" right="60" top="40" bottom="40"/>
```

In this case, the four offsets from the window borders determine the region size. If the window were 300 pixels wide by 200 pixels high, the region would be 180 pixels wide (300-60-60=180) and 120 pixels high (200-40-40=120). You could create the same layout with percentage values:

```
<region id="video_region" left="20%" right="20%" top="20%" bottom="20%"/>
```

> **Tip:** If you define a region size with these attributes, changing the window's size also changes the region's size whether the region attributes uses pixels or percentages.

Layout Example 3: Region Sizes and Two Offsets



This example shows a common way for defining region size and position. It specifies a region width and height, then sets the region's offset from the window's upper-left corner:

```
<region id="video_region" left="60" top="40" width="180" height="120"/>
```

If the window were 300 pixels wide by 200 pixels high, the region layout would be the same as in "Layout Example 2: Four Region Offsets" on page 210. Using pixel measurements for the region width and height, however, keeps the region size stable if you modify the window size.

Using Different Offset Values

For this example, you could use the `right` and `bottom` attributes instead of `left` and `top` to create the same layout:

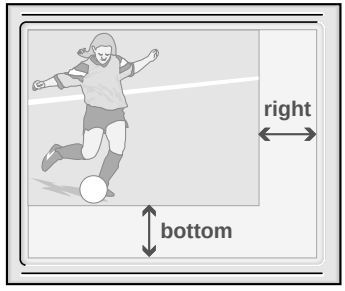
```
<region id="video_region" right="60" bottom="40" width="180" height="120"/>
```

Using Percentage Values

You could also define this layout using percentage values for the `left` and `top` offsets. This keeps the region's relative position within the window the same should you change the window size:

```
<region id="video_region" left="20%" top="20%" width="180" height="120"/>
```

Layout Example 4: Two Offsets



This example sets the region's size and position by specifying only the `right` and `bottom` attributes:

```
<region id="video_region" right="60" bottom="40"/>
```

Because neither the `left` nor the `top` attribute is defined, the region is placed in the window's upper-left corner. The region's width and height expand to meet the `right` and `bottom` offset values.

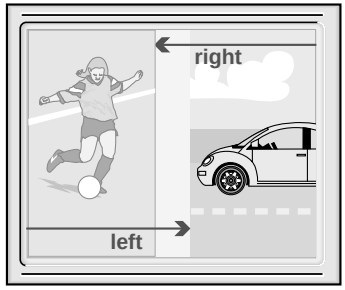
Using Different Offset Values

Alternatively, you could set the region's `left` and `top` attributes instead of `right` and `bottom` to place the region at the window's lower-right corner:

```
<region id="video_region" left="60" top="40"/>
```

Layout Example 5: Single Offsets for Two Regions

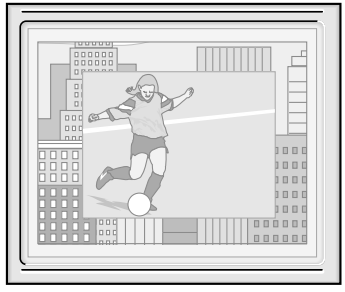
Typically, you'll need to define more that one region within a window to lay out clips that play together. To do this, you simply define each region with a separate <region/> tag, using any combination of size and position attributes to place each region in its window.



This example shows two regions laid out so that a small border of the root-layout background appears between the regions. Because vertical size or offset values (top, height, or bottom) are not specified, each region is as tall as the root-layout area:

```
<region id="region_1" right="55%"/>
<region id="region_2" left="55%"/>
```

Layout Example 6: Overlapping Regions



This example has two regions stacked on top of each other. There are many ways to define this layout with the size and position attributes. The following sample uses percentage values for the four border offsets:

```
<region id="region_1" top="5%" left="5%" bottom="5%" right="5%"/>
<region id="region_2" top="25%" left="25%" bottom="25%" right="25%"/>
```

Note: Whenever regions overlap, you should also define how the regions stack with the z-index attribute. See "Stacking Regions That Overlap" on page 214 for more information.

Tips for Defining Region Sizes and Offsets

- All regions appear within the <root-layout/> or <topLayout> area that contains them. Any part of a region defined to appear outside of its containing window is cut off. For this reason no percentage value can effectively be more than 100%.
- You can mix pixel and percentage values. You could define the top and left attributes in percentages, for example, while specifying width and height in pixels.
- If you mix pixel and percentage values when defining regions, and you also use rn:resizeBehavior="percentOnly" as described in "Changing Resize Behavior" on page 206, manually resize the RealONE Player window. If regions do not resize as expected, you may need to change some pixel values to percentages, or vice versa.
- You can use both whole and decimal values for percentages. For example, the values "4%" and "4.5%" are both valid.
- An audio clip does not require a region for playback. However, you can use a <region/> tag's soundLevel attribute to control the relative volume of an audio clip. See "Controlling Audio Volume in a Region" on page 218 for more information, and "Turning Down an Audio Clip's Volume" on page 231 for an example.

Assigning Clips to Regions

After you define the playback regions, you use region attributes within clip source tags to assign clips to regions based on the region's ID. In the following example, the video and text clips are assigned to the video and text regions defined in the header:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
 <head>
  <layout>
   <root-layout backgroundColor="maroon" width="250" height="230"/>
   <region id="video_region" top="5" left="5" width="240" height="180"/>
   <region id="text_region" top="200" left="5" width="240" height="20"/>
  </layout>
```

```
  </head>
  <body>
    <par>
      <video src="video.rm" region="video_region" .../>
      <audio src="audio.rm"/>
      <textstream src="text.rt" region="text_region" .../>
    </par>
  </body>
</smil>
```

You can reuse regions by assigning sequential clips to them. For example, you can play a video clip in a region, then display another clip in that region after the first clip finishes. You don't need to assign audio-only clips to regions at all because audio does display on the screen.

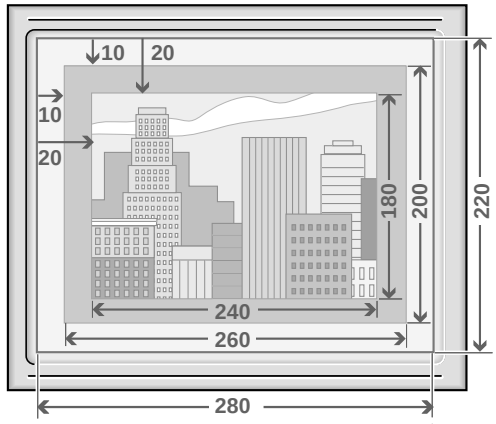
Stacking Regions That Overlap

When you define multiple regions that overlap, you can use a z-index attribute in <region/> tags to specify how regions stack. The following layout example creates a video region that overlaps an image region:

```
<layout>
  <root-layout width="280" height="220"/>
  <region id="image" top="10" left="10" width="260" height="200" z-index="0"/>
  <region id="video" top="20" left="20" width="240" height="180" z-index="1"/>
</layout>
```

In this example, the root-layout area is 220 pixels high by 280 pixels wide. The smaller image region is centered on the root-layout area. Its z-index value of 0 makes it display behind all other regions, but not behind the root-layout area. The video region centered on the image region appears on top of that region because of its higher z-index value. You could have another region overlap the video region by setting its z-index value to 2, 5, or 29, for instance. The following figure illustrates these regions.

Regions Overlapping Through z-index



Tips for Defining z-index Values

- The root-layout area always appears behind all regions. The `<root-layout/>` tag cannot have a `z-index` attribute.
- The `z-index` values can include negative integers (such as -4), 0 (zero), and positive integers (such as 5). A region with a `z-index` value of -4, for example, displays behind a region with a value of 0, which displays behind a region with a value of 5.
- The default value of 0 (zero) applies if you don't specify `z-index`.
- Using strictly sequential values such as 0, 1, 2, 3, 4 helps you keep track of the layers, but is not necessary. A sequence such as 0, 10, 20, 30, 40 works just as well, and leaving gaps in the sequence makes it easier to insert layers later.
- Nonoverlapping clips can have the same values. Side-by-side videos can both use `z-index="3"`, for example.
- When overlapping clips have the same z-index value, the clip that starts later in the presentation displays in front. If both clips start at the same time, the clip with the source tag that appears later in the SMIL file displays in front.

Adding Background Colors

By default, `<root-layout/>` and `<topLayout>` areas have a black background. All regions and subregions are transparent. In a `<root-layout/>`, `<topLayout>`, or

<region/> tag, you can specify a different background color with the backgroundColor attribute, as shown in the following example:

```
<layout>
  <root-layout backgroundColor="maroon".../>
  <region id="region1" backgroundColor="rgb(100,65,230)".../>
  <region id="region2" backgroundColor="#C2EBD7".../>
  <region id="region3" backgroundColor="inherit".../>
</layout>
```

For the color value, you can use inherit to make the region use the same color as the window or region that contains it. In the example above, the third region inherits maroon as its background color. To set a color value explicitly, use a predefined color name, a hexadecimal color value, or an RGB value.

> **For More Information:** Appendix D explains the types of color values you can use with SMIL.

> **Tip:** Using SMIL animation, you can change a region's background color as the presentation plays. See Chapter 16 for more information.

Setting When Background Colors Appear

By default, all background colors in all regions display when the presentation starts. In some cases, though, you may not want a region's background color to appear until a clip begins to play in the region. To do this, add showBackground="whenActive" to the <region/> tag:

```
<region id="region1" backgroundColor="silver" showBackground="whenActive".../>
```

Making a Region Partially Transparent

A SMIL region is fully transparent if you do not define its background color, or you explicitly set backgroundColor="transparent" in the <region/> tag. You can also make a region's background color partially transparent with the customized attribute rn:opacity="*n*%":

```
<region id="region1" backgroundColor="blue" rn:opacity="50%".../>
```

This attribute uses a percentage value from 0% (fully transparent) to 100% (fully opaque). In the example above, the value of 50% makes the region background a partially transparent blue. Using this attribute requires that you declare the following namespace in the <smil> tag:

```
xmlns:rn="http://features.real.com/2001/SMIL20/Extensions"
```

> **For More Information:** You can modify transparency in clips, too. See "Modifying Clip Colors" on page 157 for details. For background on customized attributes and namespaces, see "Using Customized SMIL Attributes" on page 139.

Transparency in Regions and Clips

If a clip that contains transparency (such as a GIF image) plays in a transparent or partially transparent region, viewers will be able see through the clip's transparent areas to underlying regions and clips. The following clip types can include transparent areas:

- RealVideo
- RealPix
- RealText
- Flash
- GIF and PNG images

RealSystem can stream other types of clips, too, and some of those clips may include transparency. Support for transparency for each clip type has to be built into RealONE Player, however. Some clips that display transparency when rendered in a Web browser, for example, may not display transparency when played in RealONE Player.

> **Tip:** To check if RealONE Player recognizes a clip's transparency, open the clip in RealONE Player and see if the window background shows through the clip's transparent areas. You can also turn a clip's background color transparent with `rn:backgroundOpacity`, as well as use `rn:mediaOpacity` to add transparency to all colors in the clip. For more on these attributes, see "Adjusting Clip Transparency and Opacity" on page 158.

Changing the Region Color Through a Clip Source Tag

By adding `backgroundColor` to a clip source tag, you can change the color of the clip's playback region. Suppose that a region uses black as a background color, and you want to play one clip in that region using a silver background instead. Rather than define a new region, you can specify the color in the clip source tag to modify the region color for as long as the clip is active:

```
<video src="..." region="video_region" backgroundColor="silver" .../>
```

Controlling Audio Volume in a Region

When a region plays a clip that includes an audio track or sound effects, you can change the clip's relative audio volume with the soundLevel attribute:

```
<region soundLevel="125%"…/>
```

The sound level always uses a percentage value. The default value of 100% plays the audio at its recorded volume. A value of 50%, for example, plays the audio at half its normal volume, whereas a value of 200% plays the audio at twice its normal volume.

Note that the soundLevel attribute controls only the relative volume of the audio stream sent to the speakers. It does not change the general sound level setting on the viewer's computer, which remains entirely under the viewer's control. All sound level adjustments are subject to limitations in the computer hardware.

> **For More Information:** See "Turning Down an Audio Clip's Volume" on page 231 for an example of how to use this attribute to change the volume of an audio clip.

Defining Subregions

A subregion functions exactly like a main region, except that it maintains its position within its containing region if you reposition the containing region. To create a subregion, you need to modify the containing region to use <region> and </region> tags instead of a single <region/> tag. You then create the subregion between the containing region's <region> and </region> tags, as shown in the following example, in which the subregion displays near the containing region's lower-right corner:

```
<head>
  <layout>
    <root-layout width="350" height="270"/>
    <region id="video_region" top="15" left="15" width="320" height="240">
      <region id="logo" bottom="5%" right="5%" width="20" height="20"/>
    </region>
  </layout>
</head>
```

You lay out a subregion within its containing region using the attributes described in "Defining Region Sizes and Positions" on page 208. When you set these attributes, keep in mind that the offset measurements of left, right, top,

and bottom are measured from the containing region's boundaries. The subregion always falls completely within the containing region.

> **For More Information:** See "Binary and Unary Tags" on page 137 for background information on converting a single <region/> tag to its binary equivalent.

Tips for Defining Subregions

- A region can hold any number of subregions.
- Subregions can be nested. A subregion can have a subregion of its own, for example.
- All subregions must have unique IDs. A subregion cannot have the same ID as another region or subregion.
- A subregion can take any <region/> tag attribute, and it does not automatically inherit any attributes from the containing region. For example, if you use fit="fill" in the containing region, and do not specify fit in the subregion, the subregion uses the default fit="hidden" rather than fit="fill".
- A subregion can have a background color the same as or different from its containing region. To keep the subregion the same color as the containing region, use backgroundColor="inherit". See "Adding Background Colors" on page 215 for more on colors.
- Because subregions always appear in front of their containing region, it is not necessary to set subregion z-index values unless multiple subregions within the containing region overlap. In this case, the subregion z-index values apply only to the subregions within the containing region. For example, suppose you define two overlapping regions, and one of these regions has two overlapping subregions:

```
<region id="regionA" z-index="1".../>
<region id="regionB" z-index="2"...>
  <region id="subregionC" z-index="1".../>
  <region id="subregionD" z-index="2".../>
</region>
```

In this example, the subregion z-index values of 1 and 2 have no relationship to the region z-index values of 1 and 2. As a result, region B appears in front of region A because it has a higher z-index value. Within region B, subregion D appears in front of subregion C.

- Creating a subregion in the layout section is useful if several clips will play in the subregion. But you can also create subregions "on the fly" within the clip source tag. See "Defining Single-Use Subregions" on page 220 for more information.

Defining Single-Use Subregions

Defining subregions in the SMIL header section is useful if you plan to reuse the subregion for multiple clips. If you want to create a subregion for just one clip, though, you can define the region in the clip source tag:

```
<layout>
  <root-layout backgroundColor="maroon" width="250" height="230"/>
  <region id="video_region" top="5" left="5" width="240" height="180"/>
</layout>
...
  <video src="video.rm" region="video_region" height="120" width="180"
   left="5" top="10"/>
...
```

In the preceding example, the video clip is assigned to an existing region, but the inclusion of `height`, `width`, `left`, and `top` values defines a single-use subregion within that region. Unlike a subregion defined in the `<layout>` section, this single-use subregion does not require an ID value. The following table lists all the region attributes that you can include in a clip source tag to set a single-use subregion's properties.

Single-Use Subregion Attributes

Attribute	Function	Reference
`backgroundColor`	Selects the subregion's background color.	page 215
`bottom`	Sets the subregion's bottom offset.	page 208
`fit`	Specifies how the clip fits its allotted space	page 227
`height`	Specifies the subregion's height.	page 208
`left`	Sets the subregion's left offset.	page 208
`regAlign`	Aligns the clip to the registration point.	page 221
`regPoint`	Defines the registration point for the subregion.	page 221
`right`	Sets the subregion's right offset.	page 208
`top`	Sets the subregion's top offset.	page 208
`width`	Specifies the subregion's width.	page 208
`z-index`	Sets stacking order relative to other subregions.	page 214

Creating Registration Points

When you do not want a clip to align with a region's upper-left corner, you can create a registration point. For example, you might define a registration point that is 10 pixels in, and 5 pixels down, from the region's upper-right corner. A registration point also includes an alignment that determines which part of the clip is placed on the point. The alignment might place the midpoint of the clip's right border on the registration point, for instance.

Within a SMIL file you can define any number of registration points using either or both of the following methods:

- Method 1: Define each registration point within each clip source tag.

  Although simpler, this method limits you to placing each clip near the center of a region, or along the region border. Plus, you must define each registration point separately in each clip tag. The section "Defining Registration Points in Clip Source Tags" on page 222 explains this method.

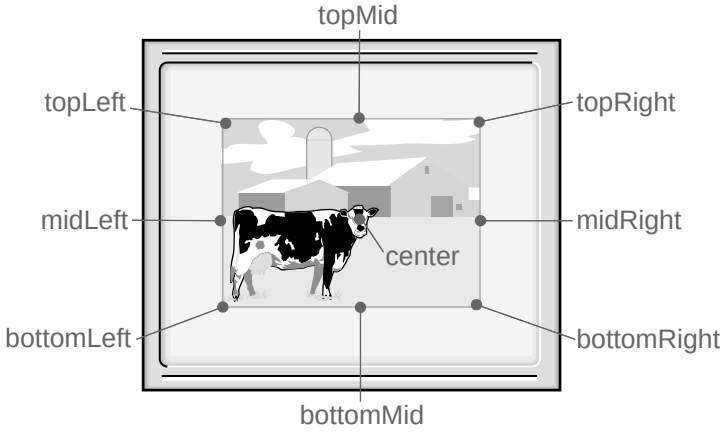
- Method 2: Define registration points with `<regPoint/>` tags in the layout section, then assign the points to clips.

  Although slightly more complex, this method is more powerful. It lets you place a clip anywhere within a region, and you can reuse each registration point in any number of clips. The section "Creating a Reusable Registration Point" on page 224 explains how to use this method.

Using Alignment Values

No matter which method you use to define registration points, you choose one of nine values to align a clip to a registration point: `topLeft`, `topMid`, `topRight`, `midLeft`, `center`, `midRight`, `bottomLeft`, `bottomMid`, or `bottomRight`. The following figure illustrates where these values fall on a clip:

Alignment Values on Clips



Defining Registration Points in Clip Source Tags

To define a registration point within a clip source tag, you add `regPoint` and `regAlign` attributes to the tag. Both attributes use an alignment value as described in the preceding section, but each value has a different meaning:

- The alignment value used with the `regPoint` attribute determines where the registration point falls in the region (hence, the alignment value applies to the region, not to the clip).
- The alignment value used with the `regAlign` attribute specifies which part of the clip aligns to the registration point.

For example, the following values center the clip in its region, regardless of the region's size and shape:

```
<ref src="..." regPoint="center" regAlign="center"/>
```

These next values select the region's lower-right corner, and place the clip's lower-right corner on that point:

```
<ref src="..." regPoint="bottomRight" regAlign="bottomRight"/>
```

Avoiding Problems When Defining Registration Points

Because you can use any of the nine predefined values for both `regPoint` and `regAlign`, there are 81 possible ways to place clips in regions using this method. Not all possibilities are useful, though. Consider this alignment:

```
<ref src="..." regPoint="topLeft" regAlign="bottomRight"/>
```

In the preceding example, `regPoint="topLeft"` puts the registration point at the region's upper-left corner. The `regAlign="bottomRight"` attribute places the clip's lower-right corner on the registration point. This locates the clip outside the region. Because a clip cannot display outside its region, the clip does not display at all.

Using Common Registration Point Values in Clip Source Tags

The following table lists some of the more useful combinations of `regPoint` and `regAlign` that you can include in a clip source tag.

Common Registration Point Values in Clip Source Tags

Clip Placement	Registration Point Values	Example
top left (default)	`regPoint="topLeft" regAlign="topLeft"`	
top center	`regPoint="topMid" regAlign="topMid"`	
top right	`regPoint="topRight" regAlign="topRight"`	
middle left	`regPoint="midLeft" regAlign="midLeft"`	
center	`regPoint="center" regAlign="center"`	
middle right	`regPoint="midRight" regAlign="midRight"`	
bottom left	`regPoint="bottomLeft" regAlign="bottomLeft"`	
bottom center	`regPoint="bottomMid" regAlign="bottomMid"`	
bottom right	`regPoint="bottomRight" regAlign="bottomRight"`	
upper-left quadrant	`regPoint="center" regAlign="bottomRight"`	
upper-right quadrant	`regPoint="center" regAlign="bottomLeft"`	
lower-left quadrant	`regPoint="center" regAlign="topRight"`	
lower-right quadrant	`regPoint="center" regAlign="topLeft"`	

Creating a Reusable Registration Point

Using the second method for creating registration points, you define each registration point in the layout section with a <regPoint/> tag. As shown in the following example, a <regPoint/> tag has a unique ID, a few positioning attributes, and a regAlign attribute:

```
<layout>
  ...windows and regions defined here...
  <regPoint id="above_center" left="50%" top="25%" regAlign="topMid"/>
</layout>
```

The preceding <regPoint/> tag creates a registration point halfway in from the left, and a quarter of the way down from the top, of any region. The regAlign value, described in "Using Alignment Values" on page 221, places the midpoint of the clip's top border on the registration point.

Positioning the Registration Point

A <regPoint/> tag's left, right, top, and bottom attributes, which can have pixel or percentage values just like region offset values, allow you to place the registration point anywhere in a region. You need to use only one or two of these attributes (such as left and top, or right and bottom) to define a registration point's position. The following table summarizes these attributes.

<regPoint/> Tag Position Attributes

Attribute	Function	Example
left	Sets the point's offset from region's left border	left="120"
right	Sets the point's offset from region's right border	right="5%"
top	Specifies the point's offset from region's top border	top="60"
bottom	Specifies the point's offset from region's bottom border	bottom="22%"

Assigning a Registration Point to Clips

Once you define a registration point in the layout section, you assign the point to any number of clips by adding a regPoint attribute to each clip source tag. This attribute takes as its value the ID of the <regPoint/> tag. For example, if you defined this registration point:

```
<regPoint id="above_center" left="50%" top="25%" regAlign="topMid"/>
```

you use the following regPoint attribute in the clip source tag:

```
<ref src="..." regPoint="above_center" .../>
```

In the clip source tag, you can even override the regAlign value defined for the registration point. Suppose that for one clip you want to use regAlign="center" instead of the defined regAlign="topMid". You can simply add the new regAlign value to the clip tag, rather than define a new registration point:

```
<ref src="..." regPoint="above_center" regAlign="center" .../>
```

> **Note:** You cannot override a registration point's position attributes, such as left and top, through a clip source tag.

Using Common Values in <regPoint/> Tags

Using <regPoint/> tags, you can replicate any registration point definable through clip source tags. The following table shows how to create common registration point alignments with values in a <regPoint/> tag rather than with attributes in clip source tags. Note that although left and top attributes are used, you could define the same registration points using right and bottom.

Common Registration Point Values in <regPoint/> Tags

Clip Placement	Registration Point Values	Example
top left (default)	left="0%" top="0%" regAlign="topLeft"	
top center	left="50%" top="0%" regAlign="topMid"	
top right	left="100%" top="0%" regAlign="topRight"	
middle left	left="0%" top="50%" regAlign="midLeft"	
center	left="50%" top="50%" regAlign="center"	
middle right	left="100%" top="50%" regAlign="midRight"	
bottom left	left="0%" top="100%" regAlign="bottomLeft"	
bottom center	left="50%" top="100%" regAlign="bottomMid"	
bottom right	left="100%" top="100%" regAlign="bottomRight"	
upper-left quadrant	left="50%" top="50%" regAlign="bottomRight"	
upper-right quadrant	left="50%" top="50%" regAlign="bottomLeft"	

(Table Page 1 of 2)

Common Registration Point Values in <regPoint/> Tags (continued)

Clip Placement	Registration Point Values	Example
lower-left quadrant	`left="50%" top="50%" regAlign="topRight"`	
lower-right quadrant	`left="50%" top="50%" regAlign="topLeft"`	

(Table Page 2 of 2)

Tips for Defining <regPoint/> Tags

- Do not use an alignment value, such as `topLeft`, as an ID in a `<regPoint/>` tag. Any variation, such as `id="alignTopLeft"` is OK, however. For information about IDs, see "SMIL Tag ID Values" on page 138.
- To keep the organization clear in the layout section, define all registration points after the `<region/>` tags.
- Keep in mind that `<regPoint/>` tags are not associated directly with `<region/>` tags. They affect regions only through the clips that play in those regions. In other words, you assign registration points to clips, and clips to regions.
- If you do not specify any position attributes, the registration point is placed in the region's upper-left corner.
- You can mix pixel and percentage values in position attributes, using `left="10"` and `top="15%"`, for example.
- Because a single registration point can apply to any region of any size, it is easier to define position attributes with percentages than with pixels.
- Because you can reuse a registration point defined in a `<regPoint/>` tag for any number of clips, it's better to use this method when you want to align many clips the same way. Once you define the `<regPoint/>` tag, you just add the single `regPoint="`*`ID`*`"` attribute to each clip tag, rather than both `regPoint="`*`value`*`"` and `regAlign="`*`value`*`"`.
- Take care not to cut off or hide clips. For example, consider these registration point attributes:

  `left="0%" top="100%" regAlign="topRight"`

  These `left` and `top` attributes place the registration point at the region's lower-left corner. The `regAlign` attribute places the clip's upper-right corner on the point. This locates the clip outside the region. Because a clip cannot display outside its region, the clip does not display at all.

- Different sizes of regions and clips, the use of registration points, and the setting of a region's fit attribute can create many different outcomes for the placement and scaling of a visual clip. For more information, get the HTML+Javascript version of this guide as described in "How to Download This Guide to Your Computer" on page 5, and choose **Clip Alignment** from the pull-down menu.

Defining How Clips Fit Regions

Whereas a registration point determines where a clip displays in a region, a fit attribute specifies what happens when a clip is larger or smaller than its allotted area. The various fit values determine whether resizing, distortion, and cropping may occur. The fit attribute is part of a `<region/>` tag, not a `<regPoint/>` tag, and it applies to a clip whether or not the clip uses a registration point. The following example shows a fit attribute set in a `<region/>` tag:

```
<region id="video_region" width="128" height="64" fit="meet"/>
```

fit Attribute Values

The fit attribute uses one of the values described in the following table. The table's last three columns indicate if the fit attribute value may scale, distort, or crop the clip if it does not fit the region dimensions exactly.

fit Value	Function	Scaling?	Distortion?	Cropping?
`fill`	Place the clip in the region's upper-left corner, or at the registration point. Scale the clip so that it fills the region exactly. Image distortion occurs if the encoded clip and playback region have different aspect ratios.	**yes**	**yes**	no
`hidden` (default)	Keep the clip at its encoded size, and place it in the region's upper-left corner, or at the registration point. If the clip is smaller than the region, fill the remaining space with the region's background color. If the clip is larger than the region, crop out the area that does not fit.	no	no	**yes**

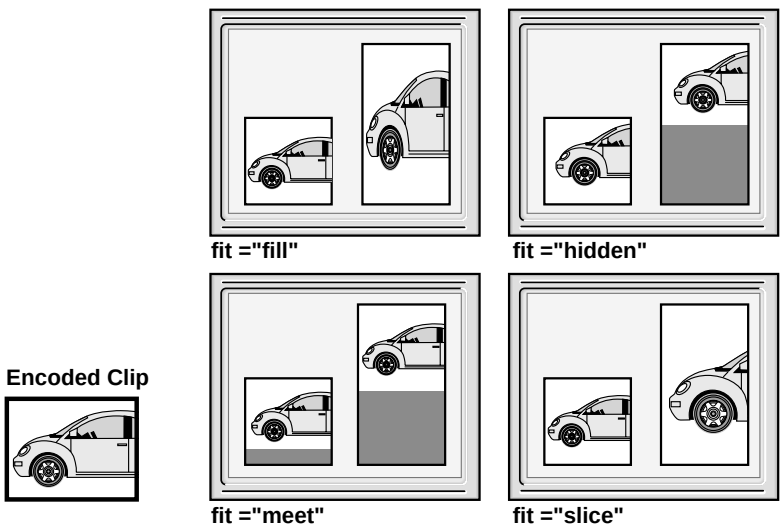
(Table Page 1 of 2)

fit Value	Function	Scaling?	Distortion?	Cropping?
`meet`	Place the clip at the region's upper-left corner or at the registration point. Scale the clip and preserve its height-to-width ratio until one dimension is equal to the region's size and the other dimension is within the region's boundaries. Fill empty space with the region's background color.	**yes**	no	no
`scroll`	Place the clip at the region's upper-left corner or at the registration point. Display the clip as its normal size, adding horizontal or vertical scroll bars if the clip extends beyond the region's boundaries. (RealPlayers earlier than RealPlayer 8 display clips as `hidden` instead.)	no	no	no
`slice`	Place the clip at the region's upper-left corner or at the registration point. Scale the clip and preserve its height-to-width ratio until one dimension is equal to the region's size and the other dimension overflows the region's boundaries. Crop the overflow.	**yes**	no	**yes**

(Table Page 2 of 2)

The following illustration shows the effects that particular fit attribute values have on a source clip played in regions with different sizes and aspect ratios.

A Clip Played in Different Regions with Different fit Attribute Values



Overriding a Region's fit Attribute

You can override a region's fit attribute within a clip source tag. Suppose that a certain region uses fit="fill", but you want to play one clip in that region and use fit="hidden" instead. You can simply add that fit value to the clip source tag to override the region's fit value:

```
<video src="..." region="textregion" fit="hidden" .../>
```

Tips for Defining the fit Attribute

- Use fit="meet" if all parts of the clip must display, if the clip's aspect ratio must be maintained, and if it's OK to scale the clip.
- Use fit="hidden" or fit="scroll" to keep the clip at its encoded size.
- Use fit="fill" if you want to fill the entire region with the clip and it doesn't matter if RealONE Player enlarges, shrinks, or distorts the clip.
- When scaling clips inside a region, keep in mind that different types of media scale with different results. A video scaled larger than its encoded size may not look good. Vector-based media such as Flash animation, on the other hand, scale more easily to fit different region sizes. Also, note that scaling a clip consumes CPU power on the RealONE Player computer.
- For recommendations on using the fit attribute with a RealText clip, see "RealText Window Size and SMIL Region Size" on page 96.
- When you use the fill, meet, or slice value, a hot spot hyperlink (image map) defined with percentage values automatically resizes with the clip. For more information, see "Choosing Percentages or Pixels for Hot Spots" on page 285.
- Different sizes of regions and clips, the use of registration points, and the setting of a region's fit attribute can create many different outcomes for the placement and scaling of a visual clip. For more information, get the HTML+Javascript version of this guide as described in "How to Download This Guide to Your Computer" on page 5, and choose **Clip Alignment** from the pull-down menu.

Layout Examples

The following sections illustrate how to use layout tags and attributes to create various types of presentations. To see more examples, get the

HTML+Javascript version of this guide as described in "How to Download This Guide to Your Computer" on page 5, and view the **Sample Files** page.

Centering a Video on a Background

This example centers a video clip on top of an image. Because region sizes are not specified, the regions expand to the root-layout size. The registration point centers the video clip within its region. The z-index attributes place the video region in front of the image region. The image region's fit="fill" attribute expands the image to fill the entire region, distorting the image if the image does not have the same aspect ratio as the region:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
 <head>
  <layout>
   <root-layout width="320" height="240"/>
   <region id="image_region" fit="fill" z-index="1"/>
   <region id="video_region" fit="hidden" z-index="2"/>
   <regPoint id="middle" regAlign="center" left="50%" top="50%"/>
  </layout>
 </head>
 <body>
  <par>
   <img src="..." region="image_region" dur="10s" fill="freeze"/>
   <video src="..." region="video_region" regPoint="middle"/>
  </par>
 </body>
</smil>
```

> **Note:** SMIL provides no way to tile an image throughout a region.

Displaying a Letterbox Clip

A widescreen movie displays on most television sets in a letterbox format, in which blank areas display above and below the movie. As shown in the following example, you can achieve the same effect for a clip that has a width-to-height ratio greater than its region's. Here, the video uses a registration point that centers it in a region that uses fit="meet" to scale the video up or down in size until its left and right edges meet the region boundaries:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
 <head>
  <layout>
   <root-layout width="400" height="300"/>
   <region id="video_region" fit="meet"/>
  </layout>
 </head>
 <body>
  <video src="..." region="video_region" regPoint="center" regAlign="center"/>
 </body>
</smil>
```

Turning Down an Audio Clip's Volume

Although audio-only clips are not typically assigned to regions, you can take advantage of a region's `soundLevel` attribute to change an audio clip's volume. The following example cuts the volume of a background music clip. The single playback region (1 pixel by 1 pixel) uses the `soundLevel` attribute to turn down the clip volume. Because the second clip is assigned to this region, RealONE Player cuts that clip's audio level as it blends it with the first clip:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
 <head>
  <layout>
   <root-layout height="1" width="1"/>
   <region id="lowvolume" soundLevel="35%"/>
  </layout>
 </head>
 <body>
  <par>
   <audio src="..." />
   <audio src="..." region="lowvolume" />
  </par>
 </body>
</smil>
```

Playing Three Clips Side-by-Side

The following example displays three regions: a news region, a video region, and a stock ticker region. The news and video regions are arranged side-by-side at the top of the RealONE Player main media window. The stock ticker region appears below them:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
 <head>
  <layout>
   <root-layout height="230" width="510" backgroundColor="black"/>
   <region id="news_region" width="240" height="180" left="5" top="5"/>
   <region id="video_region" width="240" height="180" right="5" top="5"/>
   <region id="ticker_region" width="500" height="30" left="5" bottom="5"/>
  </layout>
 </head>
 <body>
  <par endsync="id(news)">
   <textstream src="..." id="news" region="news_region" fill="freeze"/>
   <video src="..." region="video_region" fill="freeze"/>
   <textstream src="..." region="ticker_region" fill="freeze"/>
  </par>
 </body>
</smil>
```

Placing a Clip in a Secondary Media Window

A small change to the preceding example's layout can make one of the three clips display in a secondary, pop-up window. The following example places the stock ticker clip in a secondary media window that automatically opens when the presentation starts. The region that holds the stock ticker clip has no size and position information, so it automatically assumes the size of the secondary media window. The root-layout area's height has decreased, but within the SMIL body nothing has changed:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
 <head>
  <layout>
   <root-layout height="190" width="510" backgroundColor="black"/>
   <region id="news_region" width="240" height="180" left="5" top="5"/>
   <region id="video_region" width="240" height="180" right="5" top="5"/>
   <topLayout width="500" height="30">
    <region id="ticker_region"/>
   </topLayout>
  </layout>
 </head>
 <body>
  <par endsync="id(news)">
   <textstream src="..." id="news" region="news_region" fill="freeze"/>
   <video src="..." region="video_region" fill="freeze"/>
```

```
    <textstream src="..." region="ticker_region" fill="freeze"/>
   </par>
  </body>
</smil>
```

Playing the Same Clip in Multiple Regions

You normally assign a clip to a single region based on the region ID. Because each region ID must be unique, however you cannot assign the same clip to multiple regions by using region IDs. However, you can assign the same clip to two or more regions based on the region names. In the following example, the same video plays in two regions that appear size by side:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
 <head>
  <layout>
  <layout>
   <root-layout width="360" height="120"/>
   <region id="video_region1" regionName="video" soundLevel="0%"
    right="50%" fit="fill"/>
   <region id="video_region2" regionName="video" left="50%" fit="fill"/>
  </layout>
 </head>
 <body>
  <seq>
   <video src="..." region="video"/>
  </seq>
 </body>
</smil>
```

In the preceding example, the two regions share the same region name. When it reads the `<video/>` source clip tag, RealONE Player first looks for a region with `id="video"`. Because there is no such region, RealONE Player looks for a region with `regionName="video"`. In this example, it finds two regions with this name, so it plays the same clip in both regions.

With this strategy, RealONE Player requests only one video clip from RealServer. If you used two `<video/>` source clips instead, RealONE Player would request the same video stream twice, wasting bandwidth. Note, too, that one region turns off the video's audio track with the `soundLevel` attribute. If two clips have audio tracks, RealONE Player blends the tracks. In this case, that's unnecessary because the tracks are identical.

PART

V

TIMING AND LINKING CLIPS

Streaming media *flows*. Controlling when your media clips play is a crucial component for delivering a successful presentation. Chapter 12 introduces you to SMIL timing. Chapter 13 builds on that knowledge by explaining advanced timing features. To learn how to link your presentation to a Web page or another streaming presentation, read Chapter 14.

CHAPTER 12

BASIC TIMING

SMIL's timing attributes help you to tailor your presentation. You can use these attributes to adjust when clips start to play. Or you might stream just one scene from a video to create a preview without encoding a separate video clip. This chapter describes the basic SMIL timing features. Once you master these features, you can tackle advanced timing as described in Chapter 13.

Understanding Basic Timing

SMIL timing attributes are optional, giving you a powerful way to customize presentations by specifying when and how long elements play. Before you use SMIL timing attributes, though, you should know how you want to construct your overall presentation timeline. For more on this, see "Step 4: Organize the Presentation Timeline" on page 37.

> **Note:** This chapter uses the term *element* to indicate anything that can use a SMIL timing attribute. For simple presentations, elements are typically clip source tags like `<video/>` and group tags like `<par>`. But you can also use timing attributes in tags such as `<prefetch/>`, `<animate/>`, and `<area/>`.

Groups Create the Timing Superstructure

The `<seq>`, `<par>`, and `<excl>` group tags set the basic timing structure for a presentation. To stream a sequence of videos, for example, you do not need to use SMIL timing attributes. You simply arrange the clips in a `<seq>` group as described in Chapter 10. Your presentation timeline then flows automatically from the clip timelines and the group arrangement. You need to add timing attributes only if, for example, you want to add a pause between each clip, shorten the time a clip plays, or play just one scene from a clip.

Timing is Relative to Groups

In general, timing attributes for an element are relative to the group that contains the element. For elements in a <seq> group, timing attributes are relative to the end of the preceding element. For elements in a <par> or <excl> group, they're relative to the start of the group. The following example shows a sequence that consists of a parallel group followed by a video clip. For the audio clip, for example, the timing attributes are relative to the start of the <par> group:

```
<body>
 <seq ...timing is relative to the start of the presentation...>
  <par ...timing is relative to the start of the sequence...>
   <textstream ...timing is relative to the start of the parallel group.../>
   <audio ...timing is relative to the start of the parallel group.../>
  </par>
  <video ...timing is relative to the end of the preceding parallel group.../>
 </seq>
</body>
```

Timing Attributes Covered in this Chapter

The following are the basic SMIL timing attributes described in this chapter:

- begin, end, dur

  These attributes set the total length of time that an element plays. They are the most widely used of the SMIL timing attributes. See "Setting Begin and End Times" on page 240 and "Setting Durations" on page 243.

- clipBegin, clipEnd

  These attributes let you play just a portion of a clip, such as a certain scene out of a video. See "Setting Internal Clip Begin and End Times" on page 242.

- endsync

  This attribute ends a parallel or exclusive group when a certain element in the group ends. See "Ending a Group on a Specific Clip" on page 245.

- repeatCount, repeatDur

  The repeatCount and repeatDur attributes let you repeat an element a specific number of times, or for as many repetitions as possible within a certain time. See "Repeating an Element" on page 247.

- mediaRepeat

  With mediaRepeat, described in "Stopping a Clip's Encoded Repetitions" on page 249, you can stop the repetitions encoded into a clip such as an animated GIF.

- fill, erase, fillDefault

  These attributes let you keep an element visible or remove it when it is no longer active. See "Setting a Fill" on page 251 and "Specifying a Default Fill" on page 258.

Specifying Time Values

SMIL provides two methods to specify time values, a shorthand method and a "normal play time" method. Both methods provide the same capabilities. Although you can use both methods within the same SMIL file, using just one method makes authoring SMIL presentations easier.

> **Tip:** RealONE Player displays a presentation's elapsed time in one-second increments. You can click the time elapsed field to display time values to 1/10th of a second, however. This can help you decide what SMIL timing values you want to use with a clip.

Using Shorthand Time Values

The shorthand method is best suited for specifying short, simple timing values such as five seconds, ten minutes, or 1-1/2 hour. As demonstrated in the following table, the shorthand markers of h, min, s, and ms provide an easy way to designate a timing value for a SMIL element.

Timing Shorthand Markers and Examples

Timing Marker	Specifies	Example	Example Value
h	hours	end="2.5h"	2 hours, 30 minutes
min	minutes	end="2.75min"	2 minutes, 45 seconds
s	seconds	end="15.55s"	15 seconds, 550 milliseconds
ms	milliseconds	end="670.2ms"	670.2 milliseconds

> **Tip:** Decimal values are not required. You can express two seconds as "2s" or "2.0s", for example.

Using the Normal Play Time Format

The "normal play time" format for SMIL timing is suited for long, complex timing values, such as specifying one hour, fourteen minutes, 36 and 1/2 seconds. The normal play time format values use the following syntax:

```
hh:mm:ss.xy
```

where:

- hh is hours
- mm is minutes
- ss is seconds
- x is tenths of seconds
- y is hundredths of seconds

Only the ss field is required. When the time value does not include a decimal point, the last field is read as the seconds. For example, 1:30 means 1 minute and 30 seconds, whereas 1:30:00 means 1 hour and 30 minutes. Note that all of the following values are equivalent to 90 minutes:

```
begin="1:30:00.0"
begin="90:00"
begin="5400"
```

Setting Begin and End Times

The begin and end attributes affect when an element starts or stops, respectively. This section explains how to use begin and end with the basic SMIL timing values. Chapter 13 describes advanced timing values that you can use with begin and end to add interactivity to a presentation.

Using a Begin Time with a Clip

Using the begin attribute, you can vary the point at which a clip starts to play back within the presentation timeline:

```
<video src="video1.rm" begin="20.5s"/>
```

Were the preceding clip in a <par> or <excl> group, it would start playing at 20.5 seconds after the group became active. The begin attribute thereby lets you stagger the starting times of clips contained in these groups. Were this

clip in a <seq> group, there would be 20.5 seconds of blank time before the clip starts. The begin attribute therefore lets you insert delays into sequences.

> **For More Information:** See also "Setting a Fill with Sequential Clips" on page 253

Using an End Time with a Clip

You can set an end attribute alone or in combination with a begin attribute as shown in the example below, which sets the clip to end at 62.7 seconds into its part of the presentation timeline:

```
<video src="video1.rm" begin="20.5s" end="62.7s"/>
```

Note that the end time is measured from the point where the clip would start if no begin time were set. To calculate how long the clip is active, subtract the begin value from the end value. In the preceding example, the clip is active a total of 42.2 seconds (62.7 minus 20.5) regardless of the length of its internal timeline. If the clip's timeline were shorter than 42.2 seconds, the clip's last frame would display until the full 42.2 seconds had elapsed.

> **Tip:** The dur attribute gives you an alternative and sometimes simpler way to specify how long an element plays. For more information, see "Setting Durations" on page 243

Using Begin and End Times with Groups

In group tags, the begin and end attributes function much as they do in clip tags:

- If a <seq>, <par>, or <excl> group is part of a larger sequence, a begin attribute inserts "blank time" before the group becomes active. During this blank time, RealONE Player is not paused, but no activity occurs onscreen.
- If a <seq>, <par>, or <excl> group is contained in a larger <par> group, a begin value delays when the group becomes active relative to other elements in the larger <par> group.
- If a <seq>, <par>, or <excl> group is contained in a larger <excl> group, a begin value determines when it becomes active within the <excl> group.
- An end attribute in a <seq>, <par>, or <excl> group determines when the group, and hence all clips in the group, stop playing.

The following example shows a parallel group within a larger sequence. The <par> group has both a begin and an end attribute:

```
<seq>
  ...preceding elements in the sequence...
  <par begin="5s" end="3.5min">
    ...clips in the parallel group...
  </par>
  ...following elements in the sequence...
</seq>
```

In this example, the begin value delays group playback until 5 seconds after the preceding element in the sequence stops. The end attribute stops all clips in the <par> group after 3.5 minutes, regardless of their playback states. If all clips conclude before that time, there will be blank playback time before the next element in the sequence starts.

Setting Internal Clip Begin and End Times

The clipBegin and clipEnd attributes specify a clip's internal timing marks where playback begins and ends. They allow you to display just part of a clip that has an internal timeline, such as an audio, video, or animation clip. They have no effect on groups or static clips such as still images, though. The following example uses clipBegin and clipEnd with a video clip:

```
<video src="video1.rm" clipBegin="10s" clipEnd="50s"/>
```

Here, the clip starts playing at its internal 10-second mark rather than at its encoded beginning. It stops when it reaches its 50-second mark, having played for a total of 40 seconds.

> **Warning!** Do not use clipBegin and clipEnd for a live broadcast or when delivering clips with a Web server. For more information, see "Limitations on Web Server Playback" on page 428.

Combining clipBegin and clipEnd with begin and end

You can combine clipBegin and clipEnd attributes with begin and end attributes. In the following sample, a begin time is added to the preceding example:

```
<video src="video1.rm" clipBegin="10s" clipEnd="50s" begin="5s"/>
```

The begin time delays the clip's normal starting point by 5 seconds. When this time elapses, the clip starts at its 10-second internal timeline marker and plays for 40 seconds, which takes it to the 50-second mark of its internal timeline. In this case, the clipEnd attribute determines how long the video is active. You could also add an end attribute to modify this behavior, as shown in the following example:

```
<video src="video1.rm" clipBegin="10s" clipEnd="50s" begin="5s" end="50s"/>
```

Combined with the begin value, the end value of 50 means that the clip's "window of activity" within the presentation is 45 seconds. Because the clip stops playing after 40 seconds, there is an extra 5 seconds during which the clip does not play but remains active and frozen onscreen. In contrast, if you used end="30", the begin and end values would set a playback time of 25 seconds, overriding the specified clipEnd time.

Setting Durations

The dur attribute controls how long an element stays active after it starts to play. The following example ends the video after 85 seconds, regardless of the length of the clip's internal timeline. If the video's timeline is shorter than 85 seconds, the video's last frame appears frozen onscreen until the duration elapses:

```
<video src="video1.rm" dur="85s"/>
```

A common use of dur is to control how long a static clip such as an image appears onscreen. Because a static clip has an intrinsic duration of zero seconds, using dur is the easiest way to set the clip's playback time. The following example displays an image for two minutes:

```
<img src="button1.gif" dur= "2min"/>
```

Choosing end or dur

In the preceding example, end="2min" would achieve the same result as dur="2min" because no begin time is used. When a begin attribute is present, you need to calculate the end value relative to the begin value. With a dur attribute, however, you just set the total duration you want. This makes dur easier to use in some cases.

For example, suppose that you want a video to play for exactly two minutes. If a begin="20.5s" value delays the video playback for 20.5 seconds, you have to

calculate the end value by adding the begin value to the total playback time you want, as shown in the following example (140.5 - 20.5 = 120 seconds):

```
<video src="video1.rm" begin="20.5s" end="140.5s"/>
```

With a dur attribute, on the other hand, you just specify the total playback time, as shown here:

```
<video src="video1.rm" begin="20.5s" dur="2min"/>
```

As the preceding examples illustrate, you can use either end or dur depending on how you want to measure time for clip playback. For most simple timing needs, you use either end or dur, but not both. If an element includes both end and dur, the attribute that specifies the shorter playback time is used.

Setting a Duration for the Length of Media Playback

With clips that have internal timelines, you can use dur="media" to set the clip's duration to the length of its internal timeline. This is useful only when you include another timing value, such as an end attribute. Consider the following example:

```
<video src="video1.rm" end="10min" dur="media"/>
```

Because this clip uses both end and dur, the attribute specifying the shorter playback time is used. Suppose the video clip normally runs 15 minutes. In this case, end="10min" ends the clip after 10 minutes. But if the clip runs just 5 minutes, dur="media" ends the clip when it finishes its normal playback. If you did not have dur="media" in this case, the end="10min" value would keep the clip active an additional 5 minutes, unnecessarily lengthening playback.

Using an Indefinite Duration

You can use dur="indefinite" to extend an element's duration indefinitely. As with dur="media", dur="indefinite" is typically used with another timing attribute that ends the element. For example, the following clip stays active indefinitely until the viewer clicks the image with the ID stop:

```
<par>
  <ref src="..." dur="indefinite" end="stop.activateEvent" region="play"/>
  <img src="..." id="stop" dur="10s" fill="freeze" region="stopbutton"/>
</par>
```

When an element has an indefinite duration, RealONE Player's timeline slider does not operate because the presentation's end time cannot be known in advance. Hence, viewers cannot seek through the presentation. The timeline

slider will work, though, if a group timing attribute overrides the element's indefinite duration. For example, if you added dur="10min" to the <par> tag in the preceding example, RealONE Player's timeline slider would operate and indicate a presentation lasting ten minutes.

> **For More Information:** Chapter 13 explains the advanced timing commands that let you end a clip on a certain event, such as when another clip is clicked.

Tips for Setting Durations

- Like the end attribute, the dur attribute in a <seq>, <par>, or <excl> tag sets an absolute duration for the group. For more information, see "Using Begin and End Times with Groups" on page 241.
- The dur="media" and dur="indefinite" attributes are compatible with a clipBegin value. A valid clipEnd value always overrides these durations, however.
- The repeatCount attribute can play a percentage of an element when you don't know how long the element lasts. For example, you can play half of a clip by using repeatCount="0.5" instead of dur. With a three-minute clip, for example, repeatCount="0.5" is equivalent to dur="1.5min".

  > **For More Information:** For more on repeatCount, see "Repeating an Element a Certain Number of Times" on page 247.

- When you use an image in a <par> or <excl> group, you can pick a simple duration, such as dur="5s", and include fill="freeze" in the clip tag. This freezes the image until the <par> group ends, or until another element in the <excl> group replaces the image. This method is preferred over using dur="indefinite" because the indefinite value can prevent RealONE Player from determining how long the entire group lasts.

  > **For More Information:** For instructions on using the fill attribute, See "Setting a Fill" on page 251.

Ending a Group on a Specific Clip

By default, a <par> or <excl> group ends when all elements in the group finish playing. You can modify this behavior with the endsync attribute. Suppose a long clip of background music plays in parallel with a shorter RealText clip.

Using endsync, you can stop the group when the RealText clip finishes, cutting off the background music once the text has displayed. The endsync attribute has no effect in <seq> tags or clip source tags. The following table lists the endsync values.

endsync Attribute Values

Value	Function	Reference
first	Ends the group when the first clip finishes.	page 246
id(*ID*)	Ends the group when a specific clip finishes.	page 246
last	Ends the group when the last clip finishes. This is the default.	page 246

Stopping the Group After the Last Clip Finishes

The attribute endsync="last" is the default value. It causes the group to conclude when the last clip in the group stops playback. (Note that "last" refers to playback times and not the order that clips are listed in the group.) Because this is the default value, you can omit the endsync attribute from the group tag to achieve this effect.

Stopping the Group After the First Clip Finishes

Use the attribute endsync="first" to stop the group when the first clip in the group stops playing. (Note that "first" refers to playback times and not the order that clips are listed in the group.) All other clips in the group stop playing at that point, regardless of their playback statuses or any timing parameters specified for them.

Stopping the Group When a Specific Clip Finishes

The attribute endsync="id(*ID*)" causes the group to conclude when the designated element ends playback. All other elements in the group stop playing at that point, regardless of their playback statuses or any timing parameters used with them. The designated clip must have a corresponding id value in its source tag, as illustrated in the following example:

```
<par endsync="id(vid1)">
  <video id="vid1" src="..." region="video"/>
  <textstream src="..." region="text"/>
</par>
```

Tips for Using endsync Attributes

- A dur or end attribute in a <par> or <excl> tag overrides endsync. RealONE Player ends the group as specified by the dur or end attribute, not endsync.
- When you use endsync="id(*ID*)", timing attributes used with the targeted element will affect the group ending point. If the element repeats twice, for example, both repetitions must finish before the group stops.
- If you repeat the group, each repetition obeys the endsync attribute. Suppose that you define the following parallel group:

  ```
  <par endsync="first" repeatCount="2">
  ```

  The group stops when the first element stops, then repeats. On the second repetition, the group again stops when the first element stops.

Repeating an Element

Using a repeat attribute, you can specify how many times, or for how long, respectively, that an element repeats. You can also make an element repeat indefinitely. The following table summarizes these attributes.

Repeat Attributes

Attribute	Value	Function	Reference
repeatCount	*integer*|indefinite|*fractional_value*	Repeats the clip the specified number of times, or indefinitely.	page 247
repeatDur	*time_value*|indefinite	Repeats the clip the specified amount of time.	page 247

Repeating an Element a Certain Number of Times

The repeatCount attribute repeats an element a specific number of times. You can use integer values such as 2 or 4 to specify an exact number of repetitions. You can also use decimal values to stop the clip during a repetition. In the following example, the video plays 3-1/2 times:

```
<video src="video1.rm" repeatCount="3.5"/>
```

Repeating an Element a Specific Amount of Time

The repeatDur attribute repeats an element for a specified amount of time. Like a begin, end, or dur attribute, the repeatDur attribute uses a standard SMIL

timing value, as described in "Specifying Time Values" on page 239. When you use repeatDur, the element repeats as many times as it can within the specified time, shown in the following example as five minutes:

```
<video src="video1.rm" repeatDur="5min"/>
```

Specifying the Length of Each Repeating Cycle

A dur attribute included with repeatCount or repeatDur sets the total time that must elapse before the element repeats. For example, each repetition of the following clip lasts three minutes. Because the clip plays twice, the total playing time is six minutes:

```
<video src="video1.rm" repeatCount="2" dur="3min"/>
```

If the video in the preceding example has an internal timeline longer than three minutes, the video stops after three minutes and immediately repeats, playing again for just three minutes. If the video runs less than three minutes, its last frame appears frozen until the full three minutes have elapsed.

Setting a Total Playback Time

An end attribute sets the total playback time during which an element can repeat. You can use it with or without dur. For example, the repeatCount, dur, and end values in the following tag cause the clip to play one cycle in three minutes, repeat, then stop after playing a total of five minutes. This places the end of playback at two minutes into the second cycle:

```
<video src="video1.rm" repeatCount="2" dur="3min" end="5min"/>
```

With repeatDur, an end time can lengthen the time that the clip stays active after it stops repeating. In the following example, the clip repeats as many times as possible within two minutes. It then stops playing, though its last frame appears frozen onscreen for another minute:

```
<video src="video1.rm" repeatDur="2min" end="3min"/>
```

If you set end="1min" in the preceding example, the end time would override the repeatDur time of 2 minutes. There is little need to do this, however, because it's easier to set the repeatDur value to the total time you want.

Looping Playback Indefinitely

An indefinite value used with a repeatCount or repeatDur attribute causes an element to repeat until another timing attribute or user event stops the loop.

In the following example, the audio clip repeats continuously until the viewer clicks the RealONE Player **Stop** button:

```
<audio src="song.rm" repeatCount="indefinite"/>
```

As explained in "Specifying the Length of Each Repeating Cycle" on page 248, a dur attribute can set the length of each repeating cycle. In the following example, each loop lasts 30 seconds:

```
<audio src="song.rm" repeatDur="indefinite" dur="30s"/>
```

Using the indefinite value for an element in a sequence prevents the sequence from ending unless the <seq> tag itself specifies the end time with a dur or end attribute. With a <par> group, you can use endsync="id(*ID*)" to stop the group when an element other than the looping element finishes. In the following example, the audio loop stops when the RealPix slideshow concludes:

```
<par endsync="id(pix)">
  <audio src="background.rm" repeatDur="indefinite"/>
  <ref src="promo.rp" id="pix" region="images_region"/>
</par>
```

> **For More Information:** See "Ending a Group on a Specific Clip" on page 245 for more information on endsync.

Stopping a Clip's Encoded Repetitions

For clips such as animated GIF images, you can halt the clip's native repetitions by adding mediaRepeat="strip" to the clip's source tag:

```
<img src="image.gif" dur="20s" mediaRepeat="strip"/>
```

Although the mediaRepeat="strip" attribute stops a clip from repeating, it does not necessarily render a clip static. For example, an animated GIF image may consist of ten unique frames that play in sequence, with the sequence repeating indefinitely. If mediaRepeat="strip" is used, the ten unique frames play in sequence once, but do not repeat.

Once you strip out a clip's native repetitions, you can use timing attributes to set a different pattern of repetition. Suppose that a GIF image shows one frame every second for ten seconds, then repeats this cycle indefinitely. To add a delay of five seconds between each cycle, you can use the attributes shown in the following example:

```
<img src="image.gif" mediaRepeat="strip" dur="15s" repeatDur="indefinite"/>
```

In this example, the mediaRepeat attribute strips out the GIF image's native repetitions. The dur attribute sets the repeating cycle to 15 seconds, meaning the image animates as normal for 10 seconds, then pauses for five seconds. The repeatDur attribute makes this 15-second cycle repeat indefinitely.

Managing Bandwidth with Repeating Clips

When you repeat a clip streamed with RTSP or HTTP, each repetition consumes bandwidth because RealONE Player does not cache the clip. Alternatively, you can use CHTTP to cache a repeating clip on RealONE Player. The clip then consumes bandwidth only the first time it plays. You should use CHTTP only for small clips, however, because the clip cannot be larger than RealONE Player's cache size of a few Megabytes.

> **For More Information:** For more information on using CHTTP, see "Caching Clips on RealONE Player" on page 154.

Leaving Bandwidth Available for Repeating Cycles

When you stream with RTSP or HTTP, RealONE Player prebuffers each repetition to keep the presentation from pausing when the clip replays. The presentation therefore needs spare bandwidth for buffering the repeating cycles. To determine how much bandwidth to reserve, divide the clip's preroll by the amount of time that the clip plays in each cycle. Next, multiple that number by the clip's streaming bandwidth.

Suppose that a RealAudio clip streams at 20 Kbps, plays for 60 seconds, and requires 8 seconds of prebuffering. The reserve bandwidth is the following:

```
((8/60) x 20) = 2.7 Kbps
```

The inclusion of the reserve bandwidth sets the total streaming bandwidth requirement to 22.7 Kbps. This is OK for 56 Kbps modems, but too high for 28.8 Kbps modems, which have a 20 Kbps maximum as listed in the table "Maximum Streaming Rates" on page 32.

> **Tip:** To determine how much preroll a clip requires, open the clip in RealONE Player, and use the **View>Clip Source** to view clip information.

RealServer Streams Used with Repeating Clips

RealONE Player never requests more than two streams for a repeating clip. If you use repeatCount="8", for example, RealONE Player requests and plays the

first stream. As it does so, it prebuffers the second, identical stream. As it plays the second stream, it requests the first stream again, prebuffering it for the third repetition, and so on.

Tips for Repeating Elements

- Keep in mind that when an element does not repeat, `end` and `dur` both specify the clip's playing time, with the shorter value used. When an element includes `repeatDur` or `repeatCount`, though, `end` and `dur` have different functions. The `end` attribute sets the total time for all repetitions, whereas the `dur` attribute sets the length of each repeating cycle.
- With the `indefinite` value (and only the `indefinite` value), `repeatCount` and `repeatDur` function identically. Therefore, it doesn't matter if you use `repeatCount="indefinite"` or `repeatDur="indefinite"`.
- A decimal value for `repeatCount` is useful for playing just part of an element when you don't know how long the element lasts. For example, you can play half of any clip by using `repeatCount="0.5"`. With a three-minute clip, for example, `repeatCount="0.5"` is equivalent to `dur="1.5min"`.
- You can use the `clipBegin` and `clipEnd` attributes, described in "Setting Internal Clip Begin and End Times" on page 242, with repeating clips.
- When you embed a SMIL presentation in a Web page, you can use the `<EMBED>` tag's `LOOP` or `NUMLOOP` parameter to repeat the entire presentation. For more information, see "Setting Automatic Playback" on page 411.

Setting a Fill

When an element ends but is not immediately replaced by another element, you can use the `fill` attribute to specify whether the element disappears or remains onscreen. Useful primarily with visual clips and elements such as SMIL animations, the `fill` attribute does not affect audio-only clips. The following table summarizes the `fill` attribute values.

`fill` Attribute Values

Value	Function	Reference
`auto`	Makes fill behavior depend on timing attributes.	page 252
`default`	Lets `fillDefault` control the fill behavior.	page 258

(Table Page 1 of 2)

fill Attribute Values (continued)

Value	Function	Reference
freeze	Freezes element when it finishes.	page 253
hold	Keeps element visible until the group ends.	page 253
remove	Makes element disappear when it finishes.	page 253
transition	Freezes clip long enough for a transition effect to occur. This is used only with clips, and not with group tags. Chapter 15 explains the transition value.	page 336

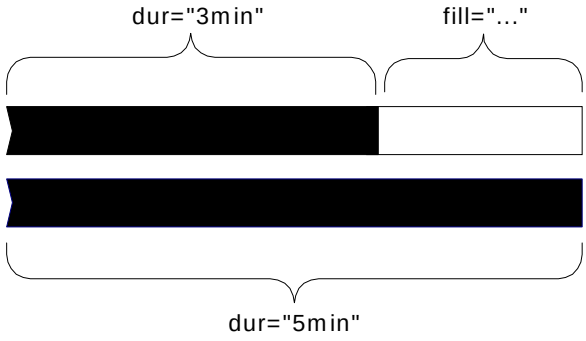
(Table Page 2 of 2)

The fill action comes after the clip's end time, as set by its internal timeline, or as specified by any timing values such as dur, end, repeatCount, or repeatDur. Consider the following example:

```
<par>
  <ref src="..." id="clip_1" region="region_1" dur="3min" fill="..."/>
  <ref src="..." id="clip_2" region="region_2" dur="5min"/>
</par>
```

The dur="3min" attribute keeps the first clip active exactly three minutes, regardless of the length of its internal timeline. The fill attribute takes effect when this duration elapses. Suppose the fill attribute freezes the clip onscreen. Because the second clip's duration makes the entire <par> group last five minutes, the first clip freezes for two minutes past its duration. The following figure illustrates this fill period.

Fill Period for Two Clips in a Parallel Group



Using an Automatic Fill

If you do not use a fill attribute with an element, and you do not set a fillDefault value in a group that contains the element, the element behaves as if

fill="auto" is set. (You can also set fill="auto" explicitly.) The effect of the auto value depends on whether certain timing elements are used:

- If the element includes a dur, end, repeatCount or repeatDur timing attribute, the fill="auto" value is equivalent to fill="remove". For example, a video that uses a dur attribute disappears when the duration expires.
- If the clip does not include any of these timing attributes, the fill="auto" value is equivalent to fill="freeze". For example, the final frame of a video that does not use any SMIL timing values freezes until the group that contains the clip ends.

> **For More Information:** The following sections explain how fill="remove" and fill="freeze" attributes affect clips in different types of groups. For more on fillDefault, see "Specifying a Default Fill" on page 258.

Setting a Fill with Sequential Clips

In a sequence of clips, a clip automatically disappears when it ends, so each clip already behaves as if it has a fill="remove" attribute. The fill="freeze" value affects a clip in a sequence only if the subsequent clip has a delayed start. In the following example, the second clip's begin time inserts a five-second delay before it plays. The fill="freeze" value keeps the first clip visible during the delay:

```
<seq>
  <video src="video1.rm" region="video_region" fill="freeze"/>
  <video src="video2.rm" region="video_region" begin="5s"/>
</seq>
```

A fill="hold" value displays a clip until the sequence ends. In the following example, an image used as a background displays first. Next, a RealText clip and video play in parallel on top of the image. Without the hold value, the image would disappear as soon as its duration elapsed. But the hold value keeps the clip visible until the entire sequence ends:

```
<seq>
  <img src="logo.gif" region="bg_region" dur="1s" fill="hold"/>
  <par>
    <textstream src="titles.rt" region="text_region" fill="freeze"/>
    <video src="video1.rm" region="video_region"/>
  </par>
</seq>
```

For the last clip in a sequence, fill="freeze" and fill="hold" function similarly. They have an effect only if the <seq> tag has a dur or end value that keeps it active after all clips have played. If all clips finish playing after eight minutes, but the <seq> tag has a dur="10min" attribute, for instance, a fill="freeze" or fill="hold" attribute for the last clip keeps that clip visible for the final two minutes of the sequence.

Setting a Fill in Parallel Groups

Use fill="remove" with a clip in a <par> group to make the clip disappear when it finishes playing. In the following example, the RealText clip disappears as soon as it finishes playing. Assuming that the video clip has a longer timeline, the parallel group ends when the video finishes playing:

```
<par>
  <textstream src="titles.rt" region="text_region" fill="remove"/>
  <video src="video1.rm" region="video_region"/>
</par>
```

In a <par> group, fill="freeze" and fill="hold" both keep a clip visible until the group completes. In the following example, the final text block of the RealText clip stays visible when the clip finishes playing. Assuming that the video clip has a longer timeline, the parallel group ends with the video clip:

```
<par>
  <textstream src="titles.rt" region="text_region" fill="freeze"/>
  <video src="video1.rm" region="video_region"/>
</par>
```

Setting a Fill in Exclusive Groups

Use fill="remove" on a clip in an <excl> group to make the clip disappear when it finishes playing. In the following example, each video clip disappears as soon as it finishes playing. If a clip finishes playing before another clip becomes active, no clip is visible on the screen:

```
<excl>
  <video src="video1.rm" region="video_region" begin="..." fill="remove"/>
  <video src="video2.rm" region="video_region" begin="..." fill="remove"/>
</excl>
```

Use fill="freeze" to keep a clip in an <excl> group visible until another clip in the group plays. Use fill="hold" to keep the clip visible until the entire <excl>

group concludes. In this case, each opaque clip needs to display in a separate region to prevent other clips from obscuring it.

Displaying a Clip Throughout a Presentation

The attribute fill="hold" keeps a clip visible only until the group that contains it ends. You can add erase="never" to fill="hold" to keep a clip visible for the entire presentation, and even after the presentation has ended. This feature, which does not work in group tags, is useful for adding a background to a presentation that contains any number of groups, as shown in the following example:

```
<body>
 <seq>
  <img src="logo.gif" region="bg_region" dur="1s" fill="hold" erase="never"/>
  ...other groups and clips...
 </seq>
</body>
```

In the preceding example, the background clip is listed as the first element in a sequence that contains other clips and groups. The fill and erase values keep the background clip visible while the subsequent clips and groups play.

Summary of Common Clip fill Values

Although the fill attribute can be used for groups and other elements such as SMIL animations, the most common use is with clips inside of groups. The following table summarizes how the most commonly used fill values affect clips that display in <seq>, <par>, and <excl> groups.

fill Attribute Values for Clips in <seq>, <par>, and <excl> Groups

Clip Attributes	Group	Function
fill="remove"	<seq>	Clip disappears when it stops playing.
	<par>	Clip disappears when it stops playing.
	<excl>	Clip disappears when it stops playing.
fill="freeze"	<seq>	Clip freezes after playback only for the duration of the subsequent clip's begin value, such as begin="5s".
	<par>	Clip freezes until the entire <par> group concludes.
	<excl>	Clip freezes until another clip in the <excl> group plays.

(Table Page 1 of 2)

fill Attribute Values for Clips in <seq>, <par>, and <excl> Groups (continued)

Clip Attributes	Group	Function
fill="hold"	<seq>	Clip freezes until the entire <seq> group concludes.
	<par>	Clip freezes until the entire <par> group concludes. Identical to fill="freeze".
	<excl>	Clip freezes until the entire <excl> group concludes.
fill="hold" erase="never"	<seq>	Clip displays throughout the presentation.
	<par>	Clip displays throughout the presentation.
	<excl>	Clip displays throughout the presentation.
fill="transition"	<seq>	Clip freezes long enough for the transition effect to occur.
	<par>	Clip freezes long enough for the transition effect to occur.
	<excl>	Clip freezes until another clip in the group plays, then remains long enough for the transition effect to occur.

(Table Page 2 of 2)

For More Information: See "Using Clip Fills with Transition Effects" on page 336 for more information on fill="transition".

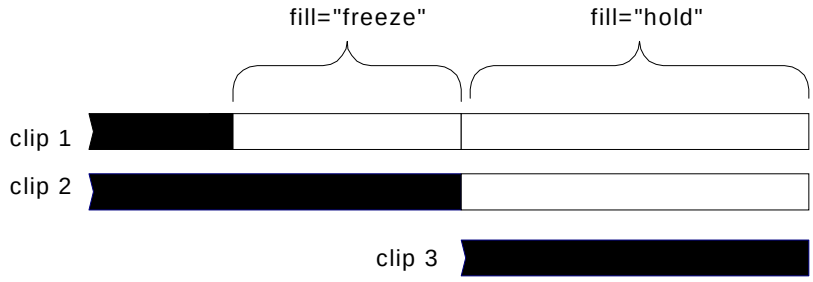
Setting a Group Fill

You can also use a fill attribute with a value of remove, freeze, or hold in a group tag. Consider the following example, in which album credits and cover art display before a song plays:

```
<seq>
  <par fill="hold">
    <img src="cover.jpg" id="clip1" region="art_region" dur="20s" fill="freeze"/>
    <textstream src="credits.rt" id="clip2" dur="60s" region="credits_region"/>
  </par>
  <audio src="song1.rm" id="clip3" dur="80s"/>
</seq>
```

In this example, the fill="freeze" value for the JPEG album cover keeps that clip visible as long as the <par> group is active. The <par> group itself has a fill="hold" value that keeps its final state visible until the containing <seq> group finishes. In other words, the <par> group's fill="hold" value extends the first two clips' fill periods until the <seq> group ends. The result is that the credits and cover image remain visible until the song completes, as the following illustration shows.

Clip Fill Periods Extended by a Group Fill



Tip: To set a fill value for a group and pass that value onto the elements in the group, use fillDefault instead of fill in the group tag.

Tips for Setting a Fill

- By default, a clip acts as if fill="freeze" is set *unless* the clip tag contains a dur, end, repeatCount or repeatDur attribute. In tags where a dur, end, repeatCount or repeatDur attribute is present, the clip acts as if fill="remove" is set. Setting a fill value explicitly, though, always overrides the default.
- In a <par> group only, you can use erase="never" with fill="freeze" to display a clip throughout the entire presentation. Because fill="hold" along with erase="never" does the same for clips in any type of group, however, it is easier always to use this latter combination.
- Using the dur attribute along with fill="remove" is the simplest means for setting how long a graphic image, which has no internal timeline, appears onscreen. In the following example, the image disappears 14.5 seconds after it appears:

```
<img src="poster.jpg" dur="14.5s" fill="remove"/>
```

- Using a short dur value along with fill="freeze" is the most common method for displaying a graphic image for as long as a parallel group is active:

```
<par>
  <img src="poster.jpg" dur="2s" fill="freeze"/>
  ...other elements in the parallel group...
</par>
```

- As long as a clip appears onscreen, any hyperlink defined for it remains active, unless the hyperlink is specifically deactivated at an earlier point. If a video links to a Web page, for example, the Web page still opens if the viewer clicks the link after the video has stopped playing and appears frozen onscreen. For more on linking, see Chapter 14.
- In SMIL 2.0, the fill attribute works slightly differently than it does in earlier versions of RealPlayer that supported SMIL 1.0. For more information, see "Behavioral Changes" on page 142.

Specifying a Default Fill

You can use the fillDefault attribute in a group tag to set a fill value for that group and its elements, whether those elements are clips or other groups. All elements within the group receive the default fill value unless they have another fill value explicitly set. The following table lists the possible fillDefault values.

fillDefault Attribute Values

Attribute	Function	Reference
auto	Makes fill behavior depend on timing attributes.	page 252
freeze	Freezes elements in the group when they finish playing.	page 253
hold	Freezes elements in the group until the group ends.	page 253
inherit	Makes each element inherit the fillDefault setting from the containing group. This is the default value.	page 259
remove	Makes elements in the group disappear when finished.	page 253
transition	Freezes clips in the group long enough for a transition effect to occur. Chapter 15 explains the transition value.	page 336

Adding a Default Fill to a Group

The following are the general rules for using fillDefault in a group tag:

- If a fillDefault value is set in a clip's group tag, and no fill value is set for the clip, the clip uses the group's fillDefault value.
- Setting a fill value explicitly in a clip source tag always makes the clip use that value regardless of any fillDefault setting in the group tag.
- A group that does not have a fillDefault value explicitly set will inherit the fillDefault value from a larger group that contains it.

The next example illustrates the fillDefault attribute set in a <par> group, with some of the group elements overriding the attribute value:

```
<par fillDefault="freeze">
  <img src="logo.gif" region="bg_region" dur="1s"/>
  <video src="video1.rm" region="video_region" fill="default"/>
  <textstream src="titles.rt" region="text_region" fill="remove"/>
</par>
```

The following fill actions occur in this group:

- The <par> group's fillDefault="freeze" value sets its fill value to freeze, and passes this value along to all its elements.
- The <img/> tag does not include a fill attribute, so it receives a fill="freeze" value from the <par> tag.
- The <video/> tag's fill="default" attribute makes it receive the freeze value just like the <img/> tag. In other words, fill="default" is the default value used with the containing group has a fillDefault attribute. (The attribute fill= "auto" is the default value if no fillDefault attribute is used.) Thus, setting fill="default" explicitly has the same effect as leaving fill out of the tag altogether.
- The <textstream/> tag includes a fill="remove" attribute, which overrides the fill="freeze" value it receives from the <par> tag.

Inheriting a Default Fill from a Containing Group

A group that does not have a fillDefault value explicitly set for it automatically inherits the fillDefault value of its containing group. The following example illustrates this inheritance with a master <par> group that contains three other <par> groups as its elements:

```
<par id="master_group" fillDefault="freeze">
  <par id="group_X">
    ...clips in group_X...
  </par>
  <par id="group_Y" fillDefault="inherit">
    ...clips in group_Y...
  </par>
  <par id="group_Z" fillDefault="remove">
    ...clips in group_Z...
  </par>
</par>
```

The following fill actions occur in this set of nested groups:

- The fillDefault="freeze" value for master_group sets the group's fill value to freeze, passing this value to all group elements.
- group_X does not include a fillDefault attribute, so it receives a fill="freeze" value from the <par> tag.
- The fillDefault="inherit" attribute in group_Y makes this group receive the freeze value from the master group. In other words, fillDefault="inherit" is the default value used with a group when its containing group has a fillDefault attribute. Setting fillDefault="inherit" explicitly has the same effect as leaving fillDefault out of the tag altogether.
- group_Z includes a fill="remove" attribute, which overrides the fillDefault="freeze" value it receives from the master group. This group uses the freeze value, passing it to all the elements it contains.

CHAPTER

13

ADVANCED TIMING

Once you have mastered the basic timing attributes described in Chapter 12, you are ready to tackle SMIL's advanced timing features. Using these features, you can develop interactive presentations that play clips when viewers click icons, for example. You can also use advanced timing to create effects similar to those found in Web pages, such as starting a SMIL animation when the viewer moves the screen pointer over an image.

> **Tip:** Be sure to familiarize yourself with "Conventions Used in this Guide" on page 9. That section lists the typographical conventions used in this chapter to explain event timing syntax.

Understanding Advanced Timing

Chapter 12 explains the basic timing attributes: `begin`, `end`, and `dur`. Although this chapter introduces some new timing attributes, it primarily shows you how to expand the power of the `begin` and `end` attributes through complex timing values. This chapter describes many different ways to start or stop an element besides using basic timing attributes such as `begin="5s"`.

Advanced Timing Syntax

The key to advanced SMIL timing is the *event*. Although not always the case, an advanced timing command typically starts or stops a SMIL element when an event occurs. So you generally have two elements that you work with: the element that triggers the event, and the element that the triggered event starts or stops. For the element that provides the event trigger, you must define an ID:

```
<element_tag1 id="ID" .../>
```

In the second element's tag, you create a `begin` or `end` value that refers to the first element's ID, specifies the triggering event, and, optionally, adds a timing offset:

```
<element_tag2 begin|end="ID.event[+|-time_value]" .../>
```

To make these abstract examples more concrete, suppose that your triggering element is a video clip:

```
<video src="video1.rm" id="intro" region="video_region"/>
```

Your triggered element might be a graphic image that begins 10 seconds after the video starts:

```
<img src="picture.jpg" begin="intro.begin+10s" region="image_region" .../>
```

In simple cases, advanced timing commands may not be needed. If the two preceding clips were in the same `<par>` group, for example, you could achieve the desired 10-second delay with simple timing commands:

```
<par>
  <video src="video1.rm" id="intro" region="video_region"/>
  <img src="picture.jpg" region="img_region" begin="10s" .../>
</par>
```

The advanced timing commands let you tie elements together when they are not in the same group, however. As well, the advanced timing commands let you start or stop clips on many kinds of events, such as mouseclicks.

Event Types

Events that can start or stop an element fall into two categories:

- scheduled events

  RealONE Player can determine that a scheduled event will happen before the event occurs. The end of a certain clip's playback is a scheduled event, for example, because RealONE Player can determine when the clip will stop based on the clip's internal timeline and the presence of SMIL timing attributes.

- interactive events

  Interactive events let you base SMIL actions on user input. But unlike a scheduled event, an interactive event such as a mouseclick cannot be known before it occurs. Some interactive events mirror scheduled events, too. The end of a clip's playback can trigger an interactive or a scheduled event, for instance.

The following table summarizes the event values you can use with the `begin` and `end` attributes. Most event values require an ID value that identifies the element that triggers the event.

begin and end Attribute Event Values

Value	Event Type	Event Trigger	Reference
`accesskey(`*`key`*`)`	interactive	keypress	page 271
`ID``.activateEvent`	interactive	mouseclick	page 269
`ID``.begin`	scheduled	beginning of element	page 266
`ID``.beginEvent`	interactive	beginning of element	page 266
`ID``.end`	scheduled	end of element	page 266
`ID``.endEvent`	interactive	end of element	page 266
`ID``.focusInEvent`	interactive	keyboard focus on element	page 271
`ID``.focusOutEvent`	interactive	keyboard focus off element	page 271
`ID``.inBoundsEvent`	interactive	pointer moving over element	page 269
`ID``.marker(`*`name`*`)`	scheduled	marker reached for element	page 274
`ID``.outOfBoundsEvent`	interactive	pointer moving off element	page 269
`ID``.repeat(`*`integer`*`)`	scheduled	specific iteration of element	page 268
`ID``.repeatEvent`	interactive	each iteration of element	page 268
`ID``.topLayoutCloseEvent`	interactive or scheduled	secondary media window closing	page 273
`ID``.topLayoutOpenEvent`	interactive or scheduled	secondary media window opening	page 273
`ID``.wallclock(`*`time`*`)`	scheduled	external clock value reached	page 274

Positive Offset Times

Most of the `begin` and `end` attribute values described in this chapter can take a positive offset timing value, which adds a delay between an event and the action that the event triggers. For example, a `begin` attribute might have the following syntax, which sets the element to start at five seconds after the event (left unspecified here) occurs:

```
begin="ID.event+5s"
```

Interactive Events with Positive Offset Times

A positive offset is useful when starting a clip based on an interactive event. Because it cannot anticipate interactive events, RealONE Player does not request clips from the server until the interactive event occurs. If you do not add a positive offset, RealONE Player may need to pause the presentation while it requests and buffers the clip's preroll. An offset such as `+15s`, on the other hand, enables RealONE Player to request the clip when the event occurs, then buffer the clip for up to 15 seconds before playing it.

> **Tip:** Instead of using a timing offset value, you can use `<prefetch/>` to request a clip's preroll in advance. For more information, see Chapter 18.

How Much of a Positive Offset Do You Need?

If you plan to start a clip on an interactive event, open the clip in RealONE Player, and use the **View>Clip Source** command to display the clip's required preroll. You'll also need to add a few seconds for RealONE Player to request the clip from the server, and to begin receiving the streamed data. If a clip's preroll is 10 seconds, for example, you may want to use positive offset of 15 seconds to ensure that the clip's preroll has streamed to RealONE Player by the time the clip begins to play.

> **Note:** Static clips such as images do not have a preroll. RealONE Player must receive all the clip data before playing the clip. The time required to display the clip is the clip size divided by the available streaming bandwidth.

Interactive Events that Do Not Require Positive Offsets

A positive offset value isn't necessary when starting or stopping elements on interactive events if those elements do not need to be streamed from a server. For example, you can use an interactive event such as a mouseclick to trigger a SMIL animation that shrinks a clip already received by RealONE Player. Because the SMIL animation is defined within the SMIL file, RealONE Player has all the data it needs to start the animation when the event occurs.

Scheduled Events with Positive Offset Times

When you start clips on scheduled events, a positive offset time is generally not required to keep the presentation flowing smoothly. RealONE Player can anticipate scheduled events and request a new clip's preroll far enough in advance to prevent presentation rebuffering. You may want to use positive

offset times with scheduled events to manage the presentation timeline, though. You might want to start a clip five seconds after another clip repeats for the second time, for example. You can do that easily by adding +5s to the clip's begin time.

Negative Offset Times

SMIL elements within a <par> or <excl> group (but not a <seq> group) can use negative timing offsets with advanced begin and end values. You can also use a negative offset value with an event, as shown in the following example:

```
begin="ID.event-5s"
```

Simple Negative Offset Times

You can use negative timing offsets in basic begin and end attributes, as well as with advanced timing commands. In the following example, the video is set to begin one minute before the group becomes active:

```
<par>
  <textstream src="credits.rt" id="credits" region="credits_region"/>
  <video src="video1.rm" region="video_region" begin="-1min"/>
</par>
```

Although the negative offset time in the preceding example is valid, a clip never plays before the group that contains it becomes active. This is because all timing attributes are relative to the group that contains the timed element. Instead of making the video clip play one minute before the parallel group becomes active, the negative offset shown above functions like clipBegin. This means that the video starts playing at its one-minute mark once the group becomes active.

For More Information: The clipBegin attribute is described in "Setting Internal Clip Begin and End Times" on page 242.

Interactive Events with Negative Offset Times

Because RealONE Player cannot anticipate an interactive event, there is no way to use a negative offset time to make a clip start or stop before an interactive event happens. If you use a negative offset to start a clip 20 seconds before an interactive event occurs, the clip begins when the event occurs, yet appears to have played for 20 seconds already. In other words, the clip acts as if clipBegin="20s" were included in its source tag.

Scheduled Events with Negative Offset Times

Negative offset values are most useful with scheduled events because RealONE Player can determine when scheduled events will occur. RealONE Player can determine when a clip is scheduled to end, for instance. You can therefore use a negative offset time to end a clip ten seconds before another clip's scheduled end time, for example.

Multiple Timing Values

For any SMIL element that uses `begin` or `end` attributes, you can define any number of timing values by separating the values with semicolons:

"*time1*; *time2*; *time3*;..."

In the following example, the clip begins when the first of two possible events occurs: either one minute elapses after the clip's group becomes active, or *event1* occurs. The clip ends either two minutes after the group starts, or when *event2* occurs:

```
<ref src="..." begin="1min; event1" end="2min; event2"/>
```

Tips for Specifying Multiple Time Values

- The order that you list time values does not matter. The time value listed third can occur before the time value listed second or first, for example.
- The entire value string must be enclosed in double quotation marks.
- You can include spaces before or after a semicolon that separates time values, but spaces are not necessary.
- Do not add a semicolon after the last value.
- The `restart` attribute can prevent a clip or group from restarting due to multiple `begin` values. See "Controlling Whether an Element Restarts" on page 274 for more information.

Defining an Element Start or Stop Event

The following four event values work with either the `begin` or the `end` attribute, letting you start or stop an element when another element begins or ends:

- *ID*.begin[+|-*time_value*]

  This scheduled event occurs when the element with the given ID begins, plus or minus any offset time. If the element repeats, this event does not occur at the start of any repeated cycles.

- *ID*.beginEvent[+|-*time_value*]

  This interactive event occurs when the element with the given ID begins, plus or minus any offset time. If the element repeats, this event occurs at the start of every cycle.

- *ID*.end[+|-*time_value*]

  This scheduled event occurs when the element with the given ID ends, plus or minus any offset time. If the element repeats, this event occurs at the end of all repeated cycles. This event does not occur if, for example, a user action stops the element before its scheduled end time.

- *ID*.endEvent[+|-*time_value*]

  This interactive event occurs when the element with the given ID ends, plus or minus any offset time. If the element repeats, this event occurs at the end of all repeated cycles. This event will not occur if the viewer stops the element by clicking the RealONE Player **Stop** button.

Sample Values

The following are samples of `begin` and `end` values that start or stop an event relative to an element with a certain ID value:

`begin="`*ID*`.end"`	Start the element when the element with the given ID is scheduled to end.
`end="`*ID*`.begin-5s"`	Stop the element five seconds before the element with the given ID is scheduled to begin.
`begin="`*ID*`.beginEvent+5s"`	Start the element five seconds after the element with the given ID actually begins.
`end="`*ID*`.endEvent"`	Stop the element when the element with the given ID actually ends.

Example

As an example of using a begin event, suppose you want to start a clip two seconds after another clip begins. You first add an ID to the element that provides the basis for starting or stopping the second element:

```
<video src="video1.rm" id="intro" region="video_region"/>
```

Next, you define the begin or end time for the second element, using the ID of the first element:

```
<img src="picture.jpg" region="img_region" dur="20s" begin="intro.begin+2s"/>
```

Keep in mind that SMIL timing values can affect when your second element begins. Suppose that the video in the preceding example has an internal timeline of two minutes, but you specify a three-minute duration as shown here:

```
<video src="video1.rm" id="intro" region="video_region" dur="3min"/>
```

If the second element uses begin="intro.end" or begin="intro.endEvent", for example, it will start to play when the video's dur time expires, which is one minute after the video displays its last frame.

Defining a Repeat Event

Two event timing values for the begin and end attributes let you start or stop a clip or group when another element repeats. You might target a specific repetition, such as the third time the element repeats. Or you can restart the clip or group on each of the element's repeating cycles:

- *ID*.repeat(*n*)[+|-*time_value*]
  This scheduled event occurs when the element with the given ID starts its specified repeating cycle, plus or minus any offset time. For example, *ID*.repeat(1) specifies the first iteration after the element has already played once.
- *ID*.repeatEvent[+|-*time_value*]
  This interactive event occurs when the element with the given ID starts its first repeated, and any subsequent, cycle.

> **Tip:** The beginEvent value occurs on every repeating cycle, including the first. See "Defining an Element Start or Stop Event" on page 266 for more information.

Sample Values

The following are samples of `begin` and `end` values that start or stop an element relative to the repetitions of another element:

`begin="ID.repeat(3)"`	Start the element when the element with the given ID begins its third repetition (that is, when it starts to play for the fourth time).
`end="ID.repeat(2)-5s"`	Stop the element five seconds before the element with the given ID begins its second repetition.
`begin="ID.repeatEvent+10s"`	Start the element ten seconds after the second and each subsequent time the element with the given ID repeats.

Example

To use a repeat timing value, you first add an ID to the clip that will provide the basis for starting or stopping the second clip. This clip must also have a `repeatCount` or `repeatDur` attribute that causes it to repeat. In the following example, the video clip repeats three times:

```
<video src="video1.rm" id="main" repeatCount="3" region="video_region"/>
```

Next, you define the `begin` or `end` time for the second clip, using the ID of the first clip:

```
<img src="picture.jpg" region="img_region" dur="20s" begin="main.repeat(2)"/>
```

> **For More Information:** For details on the `repeatCount` and `repeatDur` attributes, see "Repeating an Element" on page 247.

> **Note:** If an element repeats and has a negative timing offset, only the first cycle shows the effect of a `clipBegin`. All subsequent cycles play for their full duration.

Defining a Mouse Event

Starting or stopping an clip when a viewer clicks another clip is a common means of adding interactivity to a streaming presentation. You can also start or stop an element such as an animation when the viewer moves the screen pointer over a clip, creating an effect similar to a Javascript mouseover in a Web page. The following are the mouse-related event values that you can use with a `begin` or `end` attribute:

- *ID*.activateEvent[+|-*time_value*]

  This interactive event occurs when the viewer clicks on the clip with the specified ID. The target ID must be that of a clip, not a group or a region. The clip will not register the mouse click if it is rendered more than 50 percent transparent with a value from 0 to 50 for rn:mediaOpacity. See "Adding Transparency to All Opaque Colors" on page 158 for more information on this attribute.

- *ID*.inBoundsEvent[+|-*time_value*]

  This interactive event occurs when the viewer moves the screen pointer over the clip. The "in bounds" area is the part of the clip that displays in the region. Portions of the clip cut off at the region boundaries are not affected. The event occurs even if the clip has finished playing and appears frozen onscreen. The target ID must be that of a clip, not a group or a region.

- *ID*.outOfBoundsEvent[+|-*time_value*]

  This interactive event occurs when the viewer moves the screen pointer off of the clip's "in bounds" area. The event occurs even if the clip has finished playing and appears frozen onscreen. The target ID must be that of a clip, not a group or a region.

The inBoundsEvent and outOfBoundsEvent values can occur for multiple clips simultaneously if clips are stacked on top of each other. The z-index value of the clips does not matter, and an event can still occur even if the clip is completely obscured by another clip.

> **For More Information:** For details on z-index, see "Stacking Regions That Overlap" on page 214.

Sample Values

The following are samples of begin and end values that start or stop an element relative to a mouse event:

begin="*ID*.activateEvent"	Start the element when the clip with the given ID is clicked.
begin="*ID*.inBoundsEvent"	Start the element when the cursor moves over the clip with the given ID.
end="*ID*.outOfBoundsEvent+1s"	Stop the element one second after the cursor moves off the clip with the given ID.

Example

Suppose that you want to start a video when an image button is clicked. You first add an ID to the clip source tag of the image:

```
<img src="button1.gif" id="button" dur="5s" fill="freeze" region="button_1"/>
```

Next, you define the begin and end times for the video, using activateEvent and the image clip's ID:

```
<video src="video1.rm" region="video_region" begin="button.activateEvent"/>
```

Defining a Keyboard Event

In addition to mouse events, you can use keyboard events to start or stop elements. A keyboard event can occur when a viewer presses a key, or it can occur when a clip gains or loses the keyboard focus. When a clip has the keyboard focus, it captures all subsequent keystrokes. When a viewer clicks a form created in Flash, for example, the Flash form receives the focus. The following are begin or end event values associated with keyboard activity:

- accesskey(*key*)[+|-*time_value*]

  This interactive event occurs when the viewer presses the designated keyboard key. The key designation is case-sensitive. This value can be used along with activateEvent to provide multiple ways to start an element, either by mouseclick or keystroke.

- *ID*.focusInEvent[+|-*time_value*]

  This interactive event occurs when the clip with the designated ID receives the keyboard focus and captures subsequent keystrokes. The focus typically occurs when the viewer clicks the clip or tabs into it. The target ID must be that of a clip, not a group or a region.

- *ID*.focusOutEvent[+|-*time_value*]

  This interactive event occurs when the clip with the designated ID loses the keyboard focus. This typically occurs when the viewer clicks or tabs out of the clip. The target ID must be that of a clip, not a group or a region.

Sample Values

The following are samples of `begin` and `end` values that start or stop an element relative to a keyboard event:

`begin="accesskey(g)"`	Start the element when the keyboard letter "g" is pressed.
`end="`*`ID`*`.focusInEvent+2s"`	Stop the element two seconds after the clip with the given ID loses the keyboard focus.

Example

In the following example, the video starts playing when the keyboard letter "g" is pressed. It stops playing when the letter "h" is pressed:

```
<video src="video1.rm" region="video_region" begin="accesskey(g)"
end="accesskey(h)"/>
```

Tips for Defining Keyboard Events

- The access key value is case-sensitive, so the viewer cannot press **g** (lowercase "g") to activate the event if you specify an uppercase "G" with `accesskey(G)`, for example. You can specify both the lowercase and uppercase versions of the same key, though, to ensure that letter case does not matter.
- Access keys can be letters or numbers, but not function keys or command keys such as **Alt**, **Esc**, or **F5**.
- Mention the access key in a `longdesc` attribute in the clip source tag. See "Using a Long Description" on page 170 for more information.
- Your presentation should indicate which access keys the viewer can use. You can do this with RealText, which is described in Chapter 6. You can also display this information in the context window, as described in "Opening HTML Pages in the Context Window" on page 297.
- If the same access key is encoded into a clip to perform some function, the SMIL access key overrides the encoded key's functionality.
- You can also define access keys to open hyperlinks as described in "Opening a Link on a Keystroke" on page 291. To avoid conflicts, do not define the same key for an event and a hyperlink.

- Unlike the `inBoundsEvent` and `outOfBoundsEvent` values, which can occur for multiple clips simultaneously, only one clip at a time can have the keyboard focus at a time. Therefore, only one `focusInEvent` or `focusOutEvent` can occur at a time.

Defining a Secondary Window Event

The section "Creating Secondary Media Windows" on page 204 explains how to create a layout in which a secondary media window pops up from the main media window. The following values for the `begin` and `end` attributes allow you to start or stop an element when a secondary media window opens or closes:

- *ID*`.topLayoutOpenEvent[+|-`*time_value*`]`

  This event occurs when the secondary media window with the designated ID opens. The event is scheduled if the `<topLayout>` tag for the secondary media window uses `open="onStart"`. If the tag uses `open="whenActive"`, the window event is scheduled if the element that plays in the window has a scheduled `begin` time.

  The window event is interactive, though, if the element begins because of another interactive event. If clicking a clip in the main media window begins a clip that launches and plays in the secondary media window, for example, `topLayoutOpenEvent` is interactive.

- *ID*`.topLayoutCloseEvent[+|-`*time_value*`]`

  This event occurs when the window with the designated ID closes. The event is interactive if the `<topLayout>` tag for the secondary media window uses `close="onRequest"`. If the tag uses `close="whenNotActive"`, the window event is scheduled if the element that plays in the window has a scheduled end time.

  The window event is interactive, though, if the element ends because of another interactive event. If clicking a clip in the main media window stops the clip or clips playing in the secondary media window, for example, `topLayoutCloseEvent` is interactive.

Sample Values

The following are samples of `begin` and `end` values that start or stop an element relative to a secondary media window event:

`begin="ID.topLayoutOpenEvent"`	Start the element when the secondary media window opens.
`end="ID.topLayoutCloseEvent+2s"`	Stop the element two seconds after the secondary media window closes.

Example

The following example defines a secondary media window that opens when the first clip displays in it, and closes when all clips assigned to it finish playing:

```
<topLayout width="180" height="120" id="popup1" open="whenActive"
close="whenNotActive">
```

The following clip then starts three seconds after the window closes:

```
<video src="video1.rm" region="vid" begin="popup1.topLayoutCloseEvent+3s"/>
```

Using Media Markers

This section to be added.

Coordinating Clips to an External Clock

This section to be added.

Setting Minimum and Maximum Times

This section to be added.

Controlling Whether an Element Restarts

The `restart` attribute governs whether an element can play more than once. A clip might have multiple `begin` times that specify when it plays, for example, or start on an interactive event such as a mouse click. The `restart` attribute can

prevent an element from restarting, or place restrictions on the restart. The following table summarizes the restart values.

restart Attribute Values

Value	Function
always	Allows the element to restart at any time, even while playing. This is the effective value that is used if the element has no restart value, and no restartDefault values are specified in any groups of which the element is a member.
default	Sets the restart value to that specified by restartDefault. This is the default value that is used if no restart value is specified, but a containing group has a restartDefault value.
never	Prevents the element from restarting after it completes its first playback.
whenNotActive	Allows the element to restart only after it has completed playing. The element can then restart any number of times. The restart occurs only after the element plays to completion, its dur or end time is reached, or it finishes all of its specified repeat cycles.

In the following example, a video clip starts when a button is clicked, as described in "Defining a Mouse Event" on page 269. It uses the whenNotActive value to allow it to restart after it finishes playing. Nothing happens if the viewer clicks the activation button while the video plays. The viewer must wait for the video to stop, then click the button to restart the video:

```
<video src="video1.rm" region="video_region" begin="button.activateEvent"
restart="whenNotActive"/>
```

> **Tip:** Although the restart attribute is most commonly used with clips, you can also use it in group tags and other elements, such as SMIL animations. Keep in mind, though, that an element can restart only while its containing group is active.

Setting a Default Restart Value

You can use the restartDefault attribute in a group tag to set a restart value for the group and all of the elements it contains. All elements within the group

receive the default restart value unless they have another restart value explicitly set. The following table lists the possible restartDefault values.

restartDefault Attribute Values

Value	Function
always	Allows elements within the group to restart at any time, even while playing.
inherit	Sets the restart value for elements in the group to the restartDefault value of the group's containing group. This is the default value, meaning that a group without a restartDefault value inherits the restartDefault value from its containing group.
never	Prevents elements within the group from restarting after they complete their first playback.
whenNotActive	Allows group elements to restart any number of times, but only after they have completed playing. Restart attempts are recognized only after the elements have played to completion.

The following example shows an exclusive group of video clips in which the first two clips receive the restartDefault value of whenNotActive. The last clip, however, overrides that value with its own restart value:

```
<excl restartDefault="whenNotActive">
  <video src="video1.rm" begin="button1.activateEvent" .../>
  <video src="video2.rm" begin="button2.activateEvent" .../>
  <video src="video3.rm" begin="button3.activateEvent" restart="never" .../>
</excl>
```

Nested Group Interactions with Restart Values

If several levels of nested groups use restart and restartDefault, it's important to understand how the groups and their elements interact. Because elements inherit a restartDefault value by default, the interactions can be difficult to grasp unless you look at all levels of the nested groups. Consider the following abstract example:

```
<par id="master_group" restartDefault="whenNotActive">
  <par id="group_X" restartDefault="inherit">
    <ref id="clip_A" .../>
    <ref id="clip_B" restart="always" .../>
  </par>
  <par id="group_Y" restart="always">
    <ref id="clip_C" .../>
    <ref id="clip_D" .../>
  </par>
  <par id="group_Z" restartDefault="always">
    <ref id="clip_E" .../>
    <ref id="clip_F" restart="always" .../>
  </par>
</par>
```

The master group sets a restartDefault value of whenNotActive. The elements within this master group have the following restart values:

- group_X set to whenNotActive

  group_X inherits the default value of whenNotActive from master_group, and passes that value to the clips it contains, one of which overrides the value:

  - clip_A set to whenNotActive
  - clip_B set to always

- group_Y set to always

  group_Y sets its own behavior to always. However, it inherits the default value of whenNotActive from master_group, and passes that value to both clips it contains:

  - clip_C set to whenNotActive
  - clip_D set to whenNotActive

- group_Z set to whenNotActive

  group_Z inherits the default value of whenNotActive from master_group. However, it changes the default value for the elements it contains to always. One of the clips overrides that value:

  - clip_E set to always
  - clip_F set to whenNotActive

CHAPTER

14

HYPERLINKS

A SMIL file can define links to other media. A video might link to a second video, for example. When the viewer clicks the link, the second video replaces the first. Or the video could link to an HTML page that opens in the viewer's browser. You can even define areas as hot spots with links that vary over time. The bottom corner of a video can link to a different URL every ten seconds, for instance.

Understanding Hyperlinks

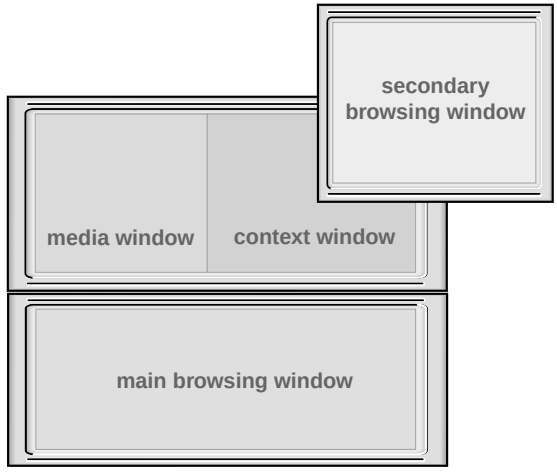
SMIL provides two hyperlink tags, both found in HTML. So if you are familiar with HTML linking, you'll pick up SMIL linking quickly. The SMIL <a> tag is the simpler means of creating hypertext links, but the <area/> tag is more powerful. The <area/> tag includes all features of <a>, and provides additional ones, such as the ability to create hot spots (image maps) and timed links. Using the <area/> tag for all hyperlinks is recommended, but the <a> tag is also available to provide basic linking functions.

> **For More Information:** The two sections "Creating a Simple Link" on page 283, and "Using the <area/> Tag" on page 283, provide the basic instructions for using the two link tags.

Links to HTML Pages

Your SMIL file can link to HTML pages that open in the RealONE Player environment (Windows only), or the viewer's default browser. The RealONE Player environment offers three types of browsing windows in addition to the viewer's default browser, as illustrated in the following figure.

Media and Browsing Windows in the RealONE Player



- media window

  The media window plays clips and SMIL presentations only. It does not display HTML page content.

- context window

  The context window can display HTML pages that supplement media clips with title and copyright information, for example. This window appears only if you explicitly open an HTML page in it through a specially configured URL.

- main browsing window

  The main browsing window can attach to, or detach from, the media and context windows. When attached, it appears below the media and context windows. Detached, it functions as a stand-alone window. You can display any HTML page associated with your presentation here.

- secondary browsing window

  RealONE Player can display any number of stand-alone, secondary browsing windows. You can display any HTML pages associated with your presentation in secondary browsing windows instead of the main browsing window.

> **Tip:** Because all of the browsing windows within the RealONE Player environment use the viewer's Internet Explorer browser

(version 4.0 or higher), they can display HTML, Javascript, and ActiveX content.

For More Information: The section "Selecting a Browsing Window" on page 295 explains the attributes that target the RealONE Player browser windows.

Links to Streaming Media

A hyperlink can also open in the RealONE Player media window, targeting an existing SMIL region, replacing the current presentation, or popping up a new media window. Note, though, that SMIL offers features that you can use in place of hyperlinking. For example, you can pop up a new window during the course of a presentation by using SMIL layout tags. With advanced timing, you can start or end a clip when the viewer clicks another clip. Neither of these features requires hyperlinks. So before you define hyperlinks, be sure that you understand the possibilities offered by SMIL.

For More Information: The section "Linking to Streaming Media" on page 301 lists the attributes and values specific to streaming media links.

Linked Pop-Up Windows vs. Secondary Pop-Up Windows

A hyperlink can pop up a new RealONE Player media window when clicked, You can also pop-up a window with a `<topLayout>` tag as described in "Secondary Media Windows" on page 197. Defining secondary media windows is appropriate when you want the new window to pop up at a predefined point in your presentation. Creating a hyperlink to a new RealONE Player media window is preferable when you want the new window to pop up based on viewer interaction, and the media you display in the window is not part of your main SMIL presentation.

Hyperlinks vs. Exclusive Groups

If you plan to create an interactive application, you need to consider carefully whether to provide interactivity through hyperlinks, `<excl>` groups, or both. Suppose that you plan to create a presentation that offers three different video clips that the viewer can select by clicking three buttons. You can author your SMIL presentation in different ways:

- Link the video clips to the buttons with hyperlinks.

  Using hyperlinks, you can link each button to a separate video. In this case, your main SMIL file does not contain <video/> tags that refer to the video clips. Instead, each button uses a hyperlink to play the clip when the button is clicked. This method works well when you want to launch each video in a separate window.

- Place the video clips in an <excl> group, and use advanced timing commands to play each video when the viewer clicks a button.

  Within an <excl> group, you include a <video/> tag for each video clip. You then use advanced SMIL timing attributes to play each video clip when the viewer clicks one of the buttons. This method is preferable if you want to display all buttons and clips in a single media window, or you want to include SMIL timing and layout attributes in each <video/> clip source tag.

  > **For More Information:** For more on the <excl> tag, see "Creating an Exclusive Group" on page 186. Chapter 13 explains advanced timing features.

Methods of Activating a Link

The screen pointer turns into a hand icon when the viewer moves the pointer over an active link in the RealONE Player media window. Typically, the viewer opens the link by clicking it. SMIL lets you define other ways to open a link, too. You might specify a keyboard key that the viewer can press to open the link, for instance. Links can also open automatically, letting you display different Web pages as a presentation plays, for example.

> **For More Information:** See "Defining Basic Hyperlink Properties" on page 290 for more information about these features.

General Tips for Creating Hypertext Tags

- Hyperlink tags work only with clip source tags. You cannot make an entire group into a hyperlink, or turn a SMIL region into an image map.
- Hyperlink tags cannot be nested. You can associate any number of hot spots or timed links defined through <area/> tags with a single clip source tag, however.

- Some clips can also define hyperlinks. A RealText clip, for example, can define hyperlinks for portions of text. When a viewer clicks an area where a clip link and a SMIL link overlap, the SMIL link is used.
- When turning a clip into a hypertext link, include the longdesc attribute in a clip source tag, using it to describe the hyperlink destination. See "Using a Long Description" on page 170 for more information.

Creating a Simple Link

The simplest type of link connects an entire source clip to another clip. As in HTML, you define the link with <a> and tags. But whereas you enclose text between <a> and in HTML, you enclose a clip source tag between <a> and in SMIL:

```
<a href="rtsp://realserver.example.com/video2.rm">
  <video src="video.rm" region="videoregion"/>
</a>
```

The preceding example links the source clip video.rm to the target clip video2.rm. When the viewer clicks video.rm as it plays, video2.rm replaces it. In an <a> tag, the href attribute is required. The URL begins with rtsp:// if the linked clip streams to RealONE Player from RealServer, or http:// if the file downloads from a Web server.

> **For More Information:** For information on link attributes, see "Defining Basic Hyperlink Properties" on page 290. See either "Linking to HTML Pages" on page 294 or "Linking to Streaming Media" on page 301 depending on your intended link target.

Using the <area/> Tag

The <area/> tag differs from the <a> tag in that you place it within the clip source tag rather than around it. This means that you must turn unary clip source tags such as <video/> into binary tags such as <video>...</video>. The <area/> tag typically ends with a closing slash, but in some cases you need to use an <area>...</area> tag pair. The following is a basic <area/> tag that links one video clip to a second video clip:

```
<video src="video.rm" region="videoregion">
  <area href="rtsp://realserver.example.com/video2.rm"/>
</video>
```

For More Information: For more on tag pairs, see "Binary and Unary Tags" on page 137. For an example of when you must use <area> and </area>, see "Setting the Context Window Size" on page 297.

If the <area/> tag includes no spatial coordinates, the entire clip becomes a link, making the <area/> tag function just like the <a> tag. A clip source tag can include any number of <area/> tags. When you define multiple <area/> links for a single clip, however, you need to do one or both of the following:

- Specify temporal coordinates so that each <area/> link is active at a different time.
- Define spatial coordinates in each <area/> tag to turn each link into a hot spot that does not overlap the other hot spots.

Creating a Timed Link

An <area/> tag can include temporal attributes that specify when the link is active, relative to the start of clip playback. If you do not include temporal attributes, the link stays active as long as the source clip appears onscreen. To add timing attributes, use the SMIL begin and end values. You cannot use dur, clipBegin, or clipEnd, however.

The following example creates two temporal links for the clip video.rm. The first link is active for the first 30 seconds of playback. The second link is active for the next 30 seconds. Because no spatial coordinates are given, the entire video is a link:

```
<video src="video.rm" region="videoregion">
  <area href="http://www.real.com" begin="0s" end="30s"…/>
  <area href="http://www.realnetworks.com" begin="30s" end="60s"…/>
</video>
```

Tip: An active link is one that the viewer can open, whether by clicking it or pressing the link's access key. The link does not open automatically, however, unless you use actuate="onLoad". For more information, see "Opening a URL Automatically" on page 293.

For More Information: See "Setting Begin and End Times" on page 240. The begin and end attributes use the SMIL timing values described in "Specifying Time Values" on page 239.

Defining Hot Spots

To create a hot spot with an <area/> tag, you use the shape attribute to define the hot spot's shape, and a coords attribute to define the hot spot's size and placement. You define the shape and coords attributes in SMIL just as you do in HTML 4.0. The following example shows two hot spots created for a clip:

```
<video src="video.rm" region="videoregion">
  <area href="..." shape="rect" coords="20,40,80,120" .../>
  <area href="..." shape="circle" coords="200,60,30" .../>
</video>
```

How you specify the coordinate values depends on what shape (rectangle, circle, or polygon) you want, as explained in the following sections. Note that in all hot spots, the coordinates are measured from the media clip's upper-left corner regardless of where you place the clip in a region.

Choosing Percentages or Pixels for Hot Spots

You can use either pixel measurements or percentages to define any hot spot. The preceding example uses pixels, whereas the following example uses percentages to place a rectangular hot spot in the center of the source clip:

```
<video src="video.rm" region="videoregion">
  <area href="..." shape="rect" coords="25%,25%,75%,75%"/>
</video>
```

When a clip is a different size than its playback region, a fit="fill", fit="meet", or fit="slice" attribute in the <region/> tag automatically resizes the clip. In these cases, a hot spot defined with percentages scales along with the clip, whereas one defined with pixels does not. Therefore, it's often better to use percentages if the region's fit attribute uses the meet, slice, or fill value. It's OK to use pixel measurements if the clip is the same size as the region, or the region's fit attribute uses the hidden or scroll value.

> **For More Information:** For more on the fit attribute values, see "Defining How Clips Fit Regions" on page 227.

> **Tip:** A viewer may also resize a presentation manually by, for example, clicking and dragging a RealONE Player corner. In these cases, hot spots scale with clips whether you define the hot spots with pixels or percentages. You can prevent a clip from resizing, though, as explained in "Changing Resize Behavior" on page 206.

Creating a Rectangular Hot Spot

Use shape="rect" to create a rectangular hot spot. You then specify four coords values in pixels or percentages to set the hot spot's size and placement, measured from the upper-left corner of the source clip in the following order:

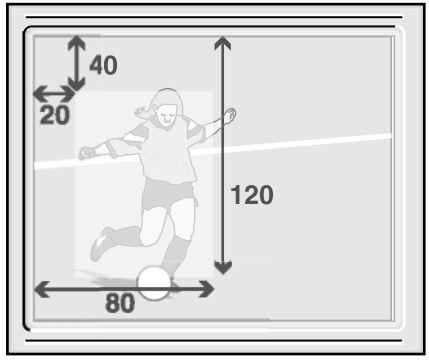
1. distance of the hot spot rectangle's left edge from the clip's left edge (left-x)
2. distance of the hot spot rectangle's top edge from the clip's top edge (top-y)
3. distance of the hot spot rectangle's right edge from the clip's left edge (right-x)
4. distance of the hot spot rectangle's bottom edge from the clip's top edge (bottom-y)

Coordinate values are separated by commas, as shown in the following example:

```
<video src="video.rm" region="videoregion">
  <area href="..." shape="rect" coords="20,40,80,120"/>
</video>
```

The preceding example uses pixel values to define a hot spot 60 pixels wide (80 pixels minus 20 pixels) and 80 pixels high (120 pixels minus 40 pixels). It creates a hot spot like that shown in the following illustration.

Rectangular Hot Spot



shape="rect" coords="20,40,80,120"

Tip: Think of the first pair of values as defining the x and y coordinates of the hot spot's upper-left corner, and the second

pair of values as defining the x and y coordinates of the hot spot's lower-right corner.

Defining a Circular Hot Spot

You can use `shape="circle"` to create a circular hot spot. Three `coords` values then specify in pixels or percentages the circle's center placement and radius in the following order:

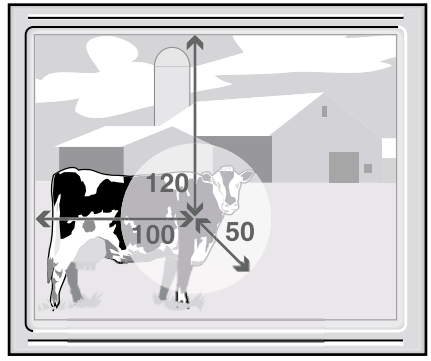
1. distance of the hot spot circle's center from clip's left edge (center-x)
2. distance of the hot spot circle's center from the clip's top edge (center-y)
3. the hot spot circle's radius

The coordinate values are separated by commas, as shown in the following example:

```
<video src="video.rm" region="videoregion">
  <area href="..." shape="circle" coords="100,120,50"/>
</video>
```

The preceding example uses pixel values to place the circular hot spot's center 100 pixels in from the clip's left edge, and 120 pixels down from the clip's top edge. The hot spot has a radius of 50 pixels. The following figure illustrates this example.

Circular Hot Spot



shape="circle" coords="100,120,50"

Tip: The last value, which sets the circle's radius, should not be more than the smaller of the other two values. If the first two values are `40` and `20`, for example, the third value should not be

more than 20. Otherwise, part of the circle extends beyond the clip boundaries and is cut off.

Making a Polygonal Hot Spot

Use shape="poly" to make a polygonal hot spot with any number of sides. You might create a triangle or an octagon, for example. For every *n* sides of the polygon you want to create, you must specify 2*n* values in the coords attribute. To create a triangle, for example, you need to specify six coords values. Each pair of coordinate values indicates the placement of a corner of the polygon in this order:

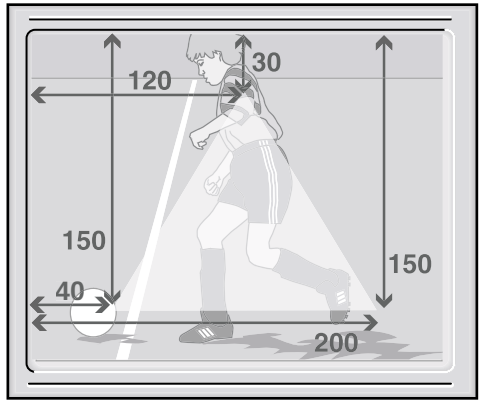
1. distance of the polygon corner from the clip's left edge (corner-x)
2. distance of the polygon corner from the clip's top edge (corner-y)

The following example defines a triangular hot spot:

```
<video src="video.rm" region="videoregion">
  <area href="..." shape="poly" coords="40,150,120,30,200,150"/>
</video>
```

The following figure illustrates the preceding example. The first value pair for the coords attribute defines the triangle's lower-left corner. The coords value pairs then proceed clockwise, defining the top corner, followed by the lower-right corner.

Polygonal Hot Spot



shape="poly" coords="40,150,120,30,200,150"

Tip: When defining a polygon, you can start with any corner, specifying the placement of additional corners by going around the polygon either clockwise or counter-clockwise.

Tips for Defining Hot Spots

- You can use whole and decimal values for percentages with the `coords` attribute. For example, the values "4%" and "4.5%" are both valid.
- You can mix pixels and percentages in the `coords` attribute. For example, the attribute `coords="50,50,100%,100%"` places a rectangular hot spot's left and top boundaries in and down 50 pixels from the source clip's upper-left corner, respectively. But the hot spot's right and bottom boundaries extend to the source clip's right and bottom edges, respectively, no matter the source clip's size.
- Values such as `coords="30,30,10,10"` for a rectangular hot spot are ignored, and the hot spot will not function. Here, the hot spot's left side is defined as being farther to the right than its right side. As well, the top is defined to be below the bottom.
- A hot spot defined to extend beyond the source clip is cropped at the clip's edge. For example, if a rectangular hot spot uses `coords="50,50,300,300"` but the source clip is 200 by 200 pixels, the hot spot's effective coordinates are `"50,50,200,200"`. For this reason, no percentage value can effectively be more than 100%.
- If multiple hot spots overlap on a clip, the link for the hot spot defined first in the SMIL file is used when the viewer clicks the overlapping area.
- Many programs, including shareware and freeware, can generate HTML image maps. You can use one of these programs to define the coordinates for a hot spot. Simply create an HTML image map over an image that is the same size as your clip, view the HTML source, and copy the image map coordinates into your `<area/>` tag.

- The following table lists sample percentage coordinates that define rectangular hot spots for a source clip. Each hot spot is a quarter the size of the source clip.

Sample Percentage Coordinates for a Rectangular Hot Spot

Hot Spot Rectangle Position	Attributes
upper-left quadrant	`shape="rect" coords="0,0,50%,50%"`
upper-right quadrant	`shape="rect" coords="50%,0,100%,50%"`
lower-left quadrant	`shape="rect" coords="0,50%,50%,100%"`
lower-right quadrant	`shape="rect" coords="50%,50%,100%,100%"`
center	`shape="rect" coords="25%,25%,75%,75%"`

Defining Basic Hyperlink Properties

The hyperlink attributes summarized in the following table affect link properties in `<a>` and `<area/>` tags whether the link opens an HTML page or a media presentation.

Basic Hyperlink Attributes

Attribute	Value	Function	Reference
`accesskey`	*key_name*	Defines a key stroke that opens the link.	page 291
`actuate`	`onLoad|onRequest`	Opens the link automatically or on request.	page 293
`alt`	*text*	Supplies alternate text.	page 293
`href`	*URL*	Provides the link URL.	page 290
`nohref`	(none)	Indicates no URL (`<area/>` tag only).	page 291
`tabindex`	*integer*	Sets a tabbing order for links.	page 294

Tip: The `accesskey`, `alt`, and `tabindex` attributes are defined the same in SMIL 2.0 as they are in HTML 4.0.

Specifying the Link URL

As with an HTML hyperlink, the SMIL `href` attribute specifies the URL to open. This should be an HTTP URL for items opened in a browser window, whether those items reside on a Web server or RealServer. SMIL files or clips opened in RealONE Player should generally have an RTSP URL if they reside

on RealServer. They must have an HTTP URL if they reside on a Web server, however. See the following sections for more information:

- For information on URL formats, see "Writing Clip Source URLs" on page 151. Although this section discusses URLs for clip source tags such as `<video/>`, the basic URL format is the same for hyperlinks.
- To display a link target in a Web browser, follow the instructions in "Linking to HTML Pages" on page 294.
- When opening a streaming media clip or SMIL file, use the additional attributes described in "Linking to Streaming Media" on page 301.

Leaving Out a URL Reference for Hot Spots

The `nohref` attribute, which can be used only in `<area/>` tags, indicates that the hot spot has no URL associated with it. You can use `nohref` with interactive timing commands to start another clip when the hot spot is clicked, for example. The `nohref` attribute does not take a value.

Opening a Link on a Keystroke

The `accesskey` attribute defines a keyboard key that the viewer can press to open the link. The viewer presses just the defined key, and does not need to press a helper key such as **Alt** to open the link. You can define any number of access keys for a link. In the following example, the viewer could press the keyboard letter **m** to open the link:

```
<area href="http://www.example.com" accesskey="m" .../>
```

Tips for Defining Access Keys

- The access key value is case-sensitive, so the viewer cannot press **m** (lowercase "m") to open the link if you specify an uppercase "M" with `accesskey="M"`, for example. You can specify both the lowercase and uppercase versions of the same key, though, to ensure that letter case does not matter.
- Access keys can be letters or numbers, but not function keys or command keys such as **Alt**, **Esc**, or **F5**.
- As long as the clip associated with the link is visible, the viewer can click the link as well as open it with the access key. You cannot define a link

that is accessible only through an access key. However, you can create the hyperlink as a very small hot spot, such as a one-pixel rectangle.

- When you make an entire source clip a link, mention the access key in a `longdesc` attribute in the clip source tag. See "Using a Long Description" on page 170 for more information.
- Your presentation should indicate which access keys the viewer can use. You can do this with RealText, which is described in Chapter 6. You can also display this information in the context window, as described in "Opening HTML Pages in the Context Window" on page 297.
- If the same access key is encoded into a clip to perform some function, the SMIL access key overrides the encoded key's functionality.
- It is best not to use the same access key when defining multiple links that are active at the same time. If multiple, active links use the same access key, the following criteria determine which link opens when the viewer presses the access key:
  - Links for clips not assigned to regions (such as audio clips) override links for clips assigned to regions.
  - When a clip displays on top of other clips because its region has a higher `z-index` value, its links override the links associated with the lower clips. However, if the upper clip uses a value of `1` to `50` for `rn:mediaOpacity`, its associated links do not register clicks, and the links for lower clips will open.

    For More Information: For more on `rn:mediaOpacity`, see "Adding Transparency to All Opaque Colors" on page 158.

  - If the `z-index` stacking order does not determine the link precedence, the link that becomes active first overrides the other links.
  - If links become active at the same time, the link listed first in the SMIL file overrides the other links.
- You can also define access keys to start or stop elements as described in "Defining a Keyboard Event" on page 271. To avoid conflicts, do not define the same key for an event and a hyperlink.

Opening a URL Automatically

The actuate attribute has a default value of onRequest, which means the link opens when the viewer clicks the link or presses the link's access key. If you set actuate="onLoad", however, the link opens as soon as the link tag becomes active in the SMIL presentation timeline, without requiring any user input. For example, the following link opens when the video clip begins to play:

```
<video src="video.rm" region="videoregion">
  <area href="http://www.example.com" actuate="onLoad"…/>
</video>
```

> **Tip:** As described in "Creating a Timed Link" on page 284, you can use a begin attribute in the <area/> tag to cause the link to become active at any point after its associated clip starts to play.

> **For More Information:** See "Displaying a Web Page when a Presentation Ends" on page 310 for an example of using actuate to open a Web page when a presentation concludes.

Displaying Alternate Link Text

A hyperlink can include an alt attribute that uses short, descriptive text as its value. It is good practice always to include an alt attribute in hyperlinks. When the viewer moves the screen pointer over the link, the alt text displays in the status line above the RealONE Player media window, indicating what the link will display. In the following example, the text "Introductory Video" is used for the alt value:

```
<area href="…" alt="Introductory Video" …/>
```

If the clip that includes the link also has an alt value, the link's alt value displays instead of the clip's. If the link has no alt value, its URL displays in place of the clip's alt value. In short, a link always overrides the clip's alt value.

> **For More Information:** The section "Including an Alternate Clip Description" on page 170 covers the alt attribute in clip source tags. See "Coded Characters" on page 167 for information on including special characters in alt text.

Setting a Tab Index for Multiple Links

When multiple links appear onscreen, the viewer can press **Tab** to cycle between the links, then press **Enter** to open a link. Using the tabindex attribute, you can specify the tabbing order. This attribute, which has a default value of 0, takes a positive integer as a value. RealONE Player highlights the clip with the lowest tabindex value first. It highlights the clip with the next higher tabindex value each time the viewer presses **Tab**. The following is an example of two clips playing in parallel, each of which has a hyperlink:

```
<par>
  <img src="..." region="ad_region"...>
    <area href="..." tabindex="2" .../>
  </img>
  <video src="..." region="video_region"... >
    <area href="..." tabindex="1" .../>
  </video>
</par>
```

In the preceding example, the link for the video clip has the lower tabindex value, so RealONE Player highlights it first when the viewer presses **Tab**. RealONE Player highlights the image clip next when the viewer presses **Tab** again.

> **Note:** If two or more <area/> tags have the same tabindex value, the tabbing order follows the order in which the clip source tags appear in the SMIL file. This also occurs if you leave tabindex out of all <area/> tags.

Linking to HTML Pages

The attributes summarized in the following table allow you to open HTML pages from your SMIL presentation. You can use these attributes to open a Web page while a presentation plays, for example. On Windows operating systems, Web page links open by default in browsing windows within the RealONE Player environment, though you can also open them in the viewer's

default browser. On other operating systems, all HTML URLs open in the viewer's default browser.

Attributes for Opening a Link in a Web Browser

Attribute	Value	Default	Function	Reference
external	true|false	false	Opens link in a browser when true.	page 295
height	*pixels*	media height	Sets context window height in <param> tag.	page 297
rn:sendTo	_osdefaultbrowser| _rpbrowser| _rpcontextwin	(none)	Specifies window that opens the HTML page.	page 295 page 297
sourceLevel	*percentage*	100%	Sets audio level.	page 306
sourcePlaystate	pause|play|stop	pause	Changes source state.	page 300
target	*name*	current window	Targets window or frame.	page 299
width	*pixels*	330	Sets context window width in <param> tag.	page 297

Selecting a Browsing Window

For a SMIL hyperlink to open in a Web browser, the external attribute must be set to true. (The external attribute's default value is false, however, which opens the link in the RealONE Player media window.) The link must also use an HTTP URL that the browser can request. Minimally, a SMIL link for content played in a Web browser looks like the following example:

```
<area href="http://www.example.com" external="true"/>
```

Using external="true" is the only requirement for opening an HTML page in a Web browser. As described in "Links to HTML Pages" on page 279, however, RealONE Player on Windows operating systems offers several browsing

windows within the RealONE Player environment. The following table lists the attributes required to open an HTML URL in one of these windows.

Attributes for Opening an HTML Link in the RealONE Player on Windows

Attributes	Target	Reference
external="true"	A secondary browsing window that does not attach to the media and context windows.	page 295
external="true" rn:sendTo="_rpbrowser"	The main RealONE browsing window, which can attach to, or detach from, the media and context windows.	page 296
external="true" rn:sendTo="_osdefaultbrowser"	The viewer's default Web browser.	page 297
external="true" rn:sendTo="_rpcontextwin"	The context window, which appears to the right of the media window.	page 297

Note the following important points about using the rn:sendTo attribute:

- The rn:sendTo attribute works only in SMIL <area/> tags. It does not function with <a> tags.
- The rn:sendTo attribute works only with RealONE Player on Windows operating systems. On Macintosh and Unix computers, the rn:sendTo attribute is ignored and the links open the viewer's default browser. This also occurs in any other SMIL-based media player.
- Using the rn:sendTo attribute requires that you declare the following namespace in the <smil> tag:

  xmlns:rn="http://features.real.com/2001/SMIL20/Extensions"

> **For More Information:** For background on customized attributes and namespaces, see "Using Customized SMIL Attributes" on page 139.

Targeting the Main Browsing Window

The RealONE Player's main browsing window can attach to, or detach from, the media and context windows. This is the recommended window for displaying Web pages along with your presentation. To target this window, declare the customized namespace in your <smil> tag as described above, and use a hyperlink that looks like the following:

<area href="http://www.example.com" external="true" **rn:sendTo="_rpbrowser"** .../>

Tip: To target this window from an HTML page displaying in the context window or a secondary browsing window within the RealONE Player environment, use `<a href="URL" target="_rpbrowser">`.

Using the Viewer's Default Browser

On Windows, Web page links open in a RealONE Player browsing window by default. Although this is preferred means for displaying these pages, you can also open these links in the viewer's default Web browser. To do this, declare the customized namespace in your `<smil>` tag as described above, and create a link that looks like the following:

```
<area href="http://www.example.com" external="true"
rn:sendTo="_osdefaultbrowser" .../>
```

Opening HTML Pages in the Context Window

Appearing to the right of the media window in the RealONE Player on Windows operating systems, the context window can display HTML pages that supplement your SMIL presentation. It might display title and copyright information about clips as they play, for example. Using advanced SMIL timing features described in Chapter 13, as well as the hyperlinking features described in this chapter, you can open a URL in the context window at any time during the presentation.

To open an HTML page in the RealONE Player context window, add `rn:sendTo="_rpcontextwin"` to the `<area/>` link tag:

```
<area href="http://www.example.com/context.html" external="true"
rn:sendTo="_rpcontextwin" sourcePlaystate="play"/>
```

Using this attribute requires that you declare the following namespace in the `<smil>` tag:

```
xmlns:rn="http://features.real.com/2001/SMIL20/Extensions"
```

Setting the Context Window Size

Through `<param/>` tags, you can extend an `<area/>` tag link to include sizing information for the RealONE Player context window. This requires that you turn your `<area/>` tag into a binary tag as described in "Binary and Unary Tags" on page 137. You can then add `<param/>` tags to the link, as shown in the following example, to specify the context window width and height in pixels:

```
<area href="..." external="true" rn:sendTo="_rpcontextwin" ...>
  <rn:param name="width" value="320"/>
  <rn:param name="height" value="240"/>
</area>
```

Tips for Using the Context Window

- The context window is designed for small HTML pages that supplement a media presentation. To display large Web pages, open URLs in a browsing window. See "Selecting a Browsing Window" on page 295.
- The context window's size is fixed for the presentation's duration. The first URL that targets the context window therefore sets the window size. Sizing information in subsequent links is ignored. The "presentation duration" includes the length of time required to play the original SMIL file, as well as any subsequent, linked clips or SMIL files.
- If a context window link does not specify a window size, a default width of 330 pixels is used. The default height is the SMIL presentation's height, which is typically set by the `height` attribute in the `<root-layout/>` tag.
- You cannot set a context window height that is smaller than the height of your presentation. If your `<root-layout/>` tag creates a presentation height of 300 pixels, for example, your context window will be 300 pixels high even if you specify a shorter height, such as 200 pixels. However, you can create a context window that is taller than your presentation, such as 400 pixels. In this case, the media window is centered vertically next to the context window.
- For best results, keep the context window approximately the same size as the media window (specified with the `<root-layout/>` tag's `height` and `width` attributes). Be careful that the combined widths of the media window and context window do not make the presentation too large to display on small computer screens.
- The media window and the context window appear next to each other with no divider. If these windows are the same height, you can create a uniform background by setting the same background color values in each window. For the media window, you set this color with the `backgroundColor` attribute of your SMIL presentation's `<root-layout/>` tag.

  For More Information: For more on SMIL background colors, see "Adding Background Colors" on page 215.

- If your context window HTML page is too large to display fully in the defined window space, the RealONE Player adds scroll bars to the context window.
- Because of the generally small size of the context window, using frames in this window is not recommended.
- The RealONE Player caches the contents of the context window URL (the HTML page, images, and so on) for the duration of the presentation. The cached contents are removed when a new presentation begins to play in the media window, however.
- To keep the presentation playing as links open in the context window, use `sourcePlaystate="play"` and `actuate="onLoad"` in the `<area/>` tag in your SMIL file.
- A standard hypertext link in a browsing window cannot open a URL in the context window because the context window requires sizing information.

Targeting a Frame or Named Window

When you use SMIL to open an HTML page, the SMIL `target` attribute works much the same as the HTML `target` attribute. When a hyperlink targets a RealONE Player secondary browsing window (using just `external="true"`) or the default browser (using `rn:sendTo="_osdefaultbrowser"`), the `target` attribute can do one of the following:

- open a new, named browsing window
- target an existing, named window
- select a named frame within an existing window

When a link specifies the main browsing window (with `rn:sendTo="_rpbrowser"`) or the context window (using `rn:sendTo="_rpcontextwin"`), the `target` attribute can select an existing frame. The following example shows how to open a link in the frame named `rightpane` within the main browsing window:

```
<area href="http://www.example.com" external="true"
rn:sendTo="_rpbrowser" target="rightpane"…/>
```

> **Tip:** The HTML values `_new` and `_top` are not supported in the RealONE Player environment. Use actual window names instead.

Controlling the Media Playback State

By default, the RealONE Player presentation pauses while the browser opens the link. The viewer can resume the presentation by clicking the RealONE Player **Play** button. RealONE Player typically needs to rebuffer the presentation briefly before continuing playback. You can also make RealONE Player stop the presentation completely, or continue playing when the link opens, with a `stop` or `play` value, respectively, for the `sourcePlaystate` attribute:

```
<area href="http://www.example.com" external="true" sourcePlaystate="play"…/>
```

The preceding link keeps the presentation playing as the browser opens the Web page. In this case, RealONE Player may need to rebuffer the presentation if browser operation uses too much bandwidth. To help prevent this, your streaming presentation should use less bandwidth than that listed in the table "Maximum Streaming Rates" on page 32.

Tips for Opening HTML Page Links

- RealONE Player passes the entire URL to the browser, which requests the resource. You can therefore include in the SMIL `href` attribute any additional parameters you want the browser to receive. A common use of this is linking to an anchor in an HTML page:

  ```
  <area href="http://www.example.com/story.html#part2" external="true"…/>
  ```

- If the RealONE Player media window is operating in full-screen mode, it resumes normal-size operation when a link opens in a browsing window. For more on full-screen mode, see "Setting a Presentation's Starting Mode" on page 422.

- If you use `sourcePlaystate="play"` to keep the SMIL presentation playing while the viewer's default Web browser opens the link, you cannot prevent the browser from obscuring RealONE Player. Whether RealONE Player remains on top of other applications as it plays is entirely under the viewer's control.

- When targeting modem users and using `sourcePlaystate="play"`, leave a few Kbps of bandwidth available to download HTML pages, depending on the size of the HTML pages that will display. To minimize bandwidth required by the browser, link to Web pages that do not contain large graphics.

- See "Adjusting Audio Volumes in Linked Presentations" on page 306 for information on using the `sourceLevel` attribute to change the RealONE Player volume when the Web page opens.

Linking to Streaming Media

When you link to another streaming media presentation, whether a SMIL file or a single clip, you can open the link URL in the existing RealONE Player media window, or pop up a new media window. The following table summarizes the attributes that you use to link to a streaming media presentation that opens in a RealONE Player media window.

Attributes for Opening a Link in the RealONE Player Media Window

Attribute	Value	Default	Function	Reference
`href="command: openwindow()"`	(*name*, *URL*)	(none)	Opens a media window from Flash or RealText.	page 306
`destinationLevel`	*percentage*	100%	Sets audio level of target.	page 306
`destination Playstate`	`pause|play`	play	Sets play state of target.	page 302
`show`	`new|replace`	replace	Opens link in a new or the current window.	page 302
`sourceLevel`	*percentage*	100%	Sets audio level of source.	page 306
`sourcePlaystate`	`pause|play| stop`	`pause|play`	Sets play state of source depending on `show`.	page 302
`target`	*ID*	(none)	Links to a specific window or region.	page 303

Replacing the Source Presentation

A link that does not include the `external="true"` attribute (which opens the link in a Web browser) replaces the current presentation in the RealONE Player media window. The source presentation is only paused, however, so the viewer can return to it by clicking RealONE Player's **Play>Previous Clip** command. Hence, an RTSP link like the following:

```
<area href="rtsp://realserver.example.com/video2.rm"/>
```

is equivalent to the following link, in which the `show`, `destinationPlaystate`, and `sourcePlaystate` attributes are explicitly set to their default values:

```
<area href="rtsp://realserver.example.com/video2.rm" show="replace"
destinationPlaystate="play" sourcePlaystate="pause"/>
```

In some cases, you may want to set `destinationPlaystate="pause"` to keep the new presentation from playing until the viewer clicks the RealONE Player **Play** button. It's not necessary ever to include the `sourcePlaystate` attribute when

replacing a presentation in RealONE Player. Its value of pause is always used with show="replace", so specifying play or stop for sourcePlaystate has no effect.

The following table summarizes the possible hyperlink attribute values for replacing a presentation in the existing RealONE Player window. The first option listed in the table is the default.

Hyperlink Attributes for Replacing a Presentation in RealONE Player

Source	Destination	Attributes
pause	play	show="replace" sourcePlaystate="pause" destinationPlaystate="play"
pause	pause	show="replace" sourcePlaystate="pause" destinationPlaystate="pause"
stop	play or pause	Not allowed. The source state is always pause.

Opening a New Media Window with SMIL

You can use either the show or the target attribute to open a new media window. The basic means for doing this is to set show="new" in the link tag. You can open any number of new windows this way. Using show="new" does not create a named window that you can target with another hyperlink, however:

```
<area href="rtsp://realserver.example.com/video2.rm" show="new".../>
```

By default, the current window containing the link and the new window with the target media are both set to play. Therefore, the preceding example is equivalent to the following example:

```
<area href="rtsp://realserver.example.com/video2.rm" show="new"
sourcePlaystate="play" destinationPlaystate="play".../>
```

Depending on how you want linking to operate, you can change the setting for sourcePlaystate to pause or stop. You can also set destinationPlaystate to pause. A common scenario is to pause the source presentation when the viewer opens the new window. The viewer can restart the source presentation by clicking the RealONE Player **Play** button. The following example illustrates this markup:

```
<area href="rtsp://realserver.example.com/video2.rm" show="new"
sourcePlaystate="pause" destinationPlaystate="play"/>
```

Targeting a Specific Window or Region

Whereas `show="new"` opens a link in a new, unnamed media window, `target="name"` creates a named window that you can select through subsequent hyperlinks. It also lets you open linked media in a specific SMIL region of an existing window, rather than in a new window. The `show="new"` attribute does not include these two capabilities.

The `target` attribute takes a user-defined name as its value. As with `show="new"`, you can set `sourcePlaystate` to `play`, `pause`, or `stop`. You can also set `destinationPlaystate` to `play` or `pause`. The following example defines a link that opens in a SMIL region or a new window named `play3`:

```
<area href="rtsp://realserver.example.com/video2.rm" target="play3"
sourcePlaystate="pause" destinationPlaystate="play".../>
```

When RealONE Player opens the link in preceding example, it displays the linked media in the following way:

1. RealONE Player displays the linked media in the existing SMIL region named `play3`. That is, it looks for a SMIL region in any open window that has the `play3` ID:

   ```
   <region id="play3" .../>
   ```

2. If no SMIL region named `play3` exists, RealONE Player displays the linked media in the window named `play3`. That is, it looks for a window created through a previous hyperlink that used a `target="play3"` attribute.

3. If no window named `play3` exists, RealONE Player creates a new window with the `play3` name, displaying the linked media in that window.

Tips for Opening Streaming Media in New Windows

- If you use both `target` and `show` in a link, the `show` attribute is ignored.
- You can use `target` exclusively to define your streaming media hyperlinks:
  - To replace the current presentation, include neither `target` nor `show`. Replacing the presentation is the default action, so you do not need to include these attributes.
  - Use `target="name"` to open a link in a SMIL region, or in a new or existing window.

- The following table summarizes the attribute values for opening streaming media in a new RealONE Player window, using either show="new" or target="*name*".

Hyperlink Attributes for Opening a Presentation in a New Window

Source	Destination	Attributes
play	play	show="new"|target="*name*" sourcePlaystate="play" destinationPlaystate="play"
play	pause	show="new"|target="*name*" sourcePlaystate="play" destinationPlaystate="pause"
pause	play	show="new"|target="*name*" sourcePlaystate="pause" destinationPlaystate="play"
pause	pause	show="new"|target="*name*" sourcePlaystate="pause" destinationPlaystate="pause"
stop	play	show="new"|target="*name*" sourcePlaystate="stop" destinationPlaystate="play"
stop	pause	show="new"|target="*name*" sourcePlaystate="stop" destinationPlaystate="pause"

- To avoid possible conflicts, use unique names for all SMIL regions and all windows that you open with target="name".
- Use short, single-word names with target="*name*".
- Do not use target="_new".
- When you open linked media in a new or existing window, the window resizes to the media's defined size.
- When you open linked media in an existing SMIL region, the window does not resize, and the region's fit attribute determines how the linked media appears if the region and media are different sizes. See "Defining How Clips Fit Regions" on page 227 for more on fit.

Linking to a SMIL Fragment

A SMIL file hyperlink can target a specific place in another SMIL file, or another part of itself. To create a link of this type, you simply include the appropriate SMIL ID in the href attribute after the URL and a pound sign (#), just as if linking to an HTML fragment:

```
<area href="rtsp://realserver.example.com/movie2.smil#text_and_video" .../>
```

The preceding link would open the designated SMIL file, and start playback at the clip or group that included the text_and_video ID:

```
<par id="text_and_video">
  <video src="video2.rm" region="newsregion"/>
  <textstream src="text.rt" region="textregion"/>
</par>
```

Note that the target SMIL file defines two regions, newsregion and textregion. When RealONE Player receives the new SMIL file, it creates those regions as specified in the new SMIL file's header.

Linking to a Clip with a Timeline Offset

You can use the <area> tag's time coordinates to create a timeline offset in a linked clip. Suppose that you want to link a video to another video at 30 seconds into the second video's timeline. In the source SMIL file, you define a link from the first video to a SMIL file that contains the second video. In the second SMIL file, the video's <area> tag defines the timeline offset using SMIL timing parameters.

Here is a sample of the link in the first SMIL file:

```
<video src="video.rm" region="videoregion">
  <area href="rtsp://realserver.example.com/newmedia.smil#vid2"/>
</video>
```

The following is the linked video clip in the second SMIL file, newmedia.smil:

```
<video src="video2.rm" region="newsregion">
  <area id="vid2" begin="30s"/>
</video>
```

For More Information: "Specifying Time Values" on page 239 describes the SMIL timing values.

Tips for Linking to SMIL Fragments

- To link to a fragment within the same SMIL file, use only href="#*ID*".
- You can link to any clip, <par>, <seq>, <excl>, or <switch> group by defining an id attribute for the clip or group. Do not link to an element in a SMIL file header, however, or to an element within a <switch> group.
- You cannot link to a clip in a <par> group and exclude the other clips in that group. All clips in the group will play in their designated regions.

- If additional clips follow the target clip in the SMIL file, those clips play as well. If you want to link to a single clip but the SMIL file that contains the clip includes other clips as well, link to the desired clip directly. Or create a new SMIL file that lists only the single target clip.

Adjusting Audio Volumes in Linked Presentations

Two attributes in a hyperlink tag, `sourceLevel` and `destinationLevel`, can change the volume of the source clip and the destination clip when a link opens. If the source clip does not stop or pause when the link opens, for example, you can use `sourceLevel` to turn down the source clip's volume and boost the destination clip's volume:

```
<a href="..." sourceLevel="35%" destinationLevel="125%" ...>...</a>
```

The audio level attributes always use a percentage value. The default value of 100% plays the audio at its recorded volume. A value of 50%, for example, plays the audio at half its normal volume, whereas a value of 200% plays the audio at twice its normal volume.

Note that the `sourceLevel` and `destinationLevel` attributes control only the relative volume of the audio stream sent to the speakers. They do not change the general sound level setting on the viewer's computer, which remains entirely under the viewer's control. All sound level adjustments are subject to limitations in the computer hardware.

> **Tip:** When displaying a Web page, as described in "Linking to HTML Pages" on page 294, you can use `sourceLevel` to turn down or boost RealONE Player's volume as appropriate. The `destinationLevel` attribute will not affect any audio elements, such as an embedded WAV file played by the browser, though.

Opening a New Media Window Through RealText or Flash

A RealText or a Flash clip, playing alone or as part of a SMIL presentation, can define a hyperlink that opens another clip in a new media window. This link uses a proprietary parameter, `command:openwindow(`*`name`*`,`*`URL`*`)`, as the value of the `href` attribute. This is not a SMIL feature, and you write this parameter directly into the RealText markup, or encode it in the Flash Player file with the **Get URL** command.

The hypertext reference for this type of link has the following structure:

```
href="command:openwindow(name, URL, [zoomlevel=double|full|normal])"
```

The command:openwindow parameter requires two arguments, *name* and *URL*. The zoomlevel argument is optional. You can separate arguments with a comma, but this is not required. A space may precede or follow a comma. If an argument contains characters such as commas or parentheses, enclose it in single quotation marks.

Window Names

The required *name* argument, which supplies a predefined or user-defined name for the new media window, is the first parameter listed for command:openwindow. The following table describes the parameter values.

name Parameter

Value	Action
_new or _blank	Opens a new media window each time the viewer clicks the link. Each subsequent link named _new or _blank opens a new window as well.
_self or _current	Opens the URL in the current media window.
name	Creates a new media window with the user-defined name. A subsequent openwindow command using the same name opens the given URL in the same window.

Target URL

Following the *name* argument, the required *URL* argument gives the fully qualified URL to the clip or SMIL presentation to play in the new window. You must include the protocol (rtsp://, http://, chttp://, or file://) in the URL. Relative URLs do not work.

Zoom Level

The optional zoomlevel=double|full|normal argument sets the new media window to open in double-size or full-screen mode respectively. The normal value is the default. If the operating system does not support full-screen mode, normal mode is used instead.

> **Tip:** You can also open the initial presentation in double or full-screen mode by using a Ram file. For details on doing this, as well as guidelines for using double and full-screen modes, see "Setting a Presentation's Starting Mode" on page 422.

Note: Earlier versions of RealPlayer support additional parameters, such as autosize and ontopwhileplaying, that RealONE Player ignores. RealONE Player is therefore backwards-compatible with presentations developed for earlier versions of RealPlayer. These additional parameters are obsolete, however.

Examples

The following examples show how to target various windows with the command:openwindow hyperlink syntax inRealText and Flash clips.

Targeting the Same Window with Multiple Links

The following RealText link opens a URL in a new media window named feature:

```
<a href="command:openwindow(feature,
rtsp://realserver.example.com/comedy.rm">Comedy Hour</a>
```

In Flash, the **Get URL** command looks like this:

```
command:openwindow(feature, rtsp://realserver.example.com/comedy.rm)
```

When first clicked, this link creates a media window named feature. If another link also targets the feature window, clicking that link starts the new URL in the feature window. Clicking the link in the following example starts an animal program in the window running the comedy program:

```
<a href="command:openwindow(feature,
rtsp://realserver.example.com/animals.rm)">Sharks!</a>
```

The Flash **Get URL** version looks like this:

```
command:openwindow(feature, rtsp://realserver.example.com/animals.rm)
```

Opening Separate Windows

Each link opens a separate window if the window names are different, or you use the predefined name _new or _blank. The following RealText links open separate windows:

```
<a href="command:openwindow(_new,
rtsp://realserver.example.com/comedy.rm)">Comedy Hour</a>
```

```
<a href="command:openwindow(_blank,
rtsp://realserver.example.com/animals.rm,)">Sharks!</a>
```

In Flash, the **Get URL** commands look like these:

```
command:openwindow(_new, rtsp://realserver.example.com/comedy.rm)
```

```
command:openwindow(_blank, rtsp://realserver.example.com/animals.rm)
```

Launching Clips in the Current Window

Use either _current or _self to open the URL in the current window. The following example is for RealText:

```
<a href="command:openwindow(_self,
rtsp://realserver.example.com/comedy.rm)">Comedy Hour</a>
```

The next RealText link plays the clip at double its encoded size:

```
<a href="command:openwindow(_current,
rtsp://realserver.example.com/animals.rm, zoomlevel=double)">Sharks!</a>
```

The following are the same commands issued through **Get URL** in Flash:

```
command:openwindow(_self, rtsp://realserver.example.com/comedy.rm)
```

```
command:openwindow(_current, rtsp://realserver.example.com/animals.rm,
zoomlevel=double)
```

Tips for Opening Media Windows with RealText or Flash

- Unlike HTML, RealSystem markup tags are case-sensitive. Be sure to use lowercase for `command:openwindow` and its parameters.
- When the viewer clicks a `command:openwindow` link, the new clip automatically plays, and the presentation that contains the link stops. You cannot change this playback state to pause the original presentation, for example.
- RealText and Flash clips can also open links in a Web browser. For basic information on hypertext links in RealText, see "Creating Links and Issuing Commands" on page 116.
- Because `command:openwindow` is not a SMIL command, it does not offer all the SMIL linking features, such as activating automatically or on a keystroke.
- If a SMIL hyperlink overlaps a `command:openwindow` link, the SMIL link takes precedence.

Hyperlink Examples

The following examples show different applications of hyperlinking. To see more examples, get the HTML+Javascript version of this guide as described in

"How to Download This Guide to Your Computer" on page 5, and view the **Sample Files** page.

Displaying a Web Page when a Presentation Ends

Using the <area/> tag, you can easily display Web pages as a clip plays. The following example uses advanced SMIL timing commands to open a Web page when the video clip ends:

```
<video src="video1.rm" id="main" fill="freeze">
  <area href="http://www.example.com" external="true" rn:sendTo="_rpbrowser"
    actuate="onLoad" begin="main.end"/>
</video>
```

In this example, the hyperlink uses actuate="onLoad" to open the URL automatically when the link becomes active. The begin="main.end" attribute opens the link when the clip with id="main" ends. This advanced timing feature is described in "Defining an Element Start or Stop Event" on page 266.

The rn:sendTo attribute sends the URL to the RealONE Player main browsing window, as described in "Targeting the Main Browsing Window" on page 296. You can leave this attribute out of the <area/> tag to open the URL in a secondary browsing window. Including the rn:sendTo attribute requires that you declare the following namespace in the <smil> tag:

```
xmlns:rn="http://features.real.com/2001/SMIL20/Extensions"
```

Opening Web Pages During a Presentation

The following markup uses a series of <area/> tags with different begin times to open four Web pages at different points as an audio clip plays. The actuate="onLoad" attribute causes each link to open its Web page as soon as the link becomes active. Because the links do not use rn:sendTo="_rpbrowser" as in the preceding example, the pages open in secondary browsing windows. The attribute sourcePlaystate="play" keeps the clip playing as each page opens:

```
<audio src="audio1.rm">
  <area href="http://www.example.com/page1.htm" begin="30s" external="true"
    actuate="onLoad" sourcePlaystate="play"/>
  <area href="http://www.example.com/page2.htm" begin="1min" external="true"
    actuate="onLoad" sourcePlaystate="play"/>
  <area href="http://www.example.com/page3.htm" begin="2min" external="true"
```

```
   actuate="onLoad" sourcePlaystate="play"/>
  <area href="http://www.example.com/page4.htm" begin="3min" external="true"
   actuate="onLoad" sourcePlaystate="play"/>
</audio>
```

Tip: Opening a Web page requires bandwidth. If your streaming media uses all of the viewer's available bandwidth, opening a Web page may cause the presentation to stall. When opening Web pages during a presentation, be sure that your streaming media uses less bandwidth than the maximum listed in the table "Maximum Streaming Rates" on page 32.

PART

VI

MASTERING ADVANCED SMIL FEATURES

With the basics mastered, you're ready to learn SMIL's power features. Chapter 15 explains how to create special effects when a clip starts or stops playing. Read Chapter 16 to learn how to transform clips as they play. Chapter 17 explains how to stream different clips based on viewer criteria, such as language preference. Prefetching, which Chapter 18 describes, lets you download clip data before a clip plays.

CHAPTER

15

TRANSITION EFFECTS

You can enhance your presentation's appeal by adding visual effects that occur when any type of visual clip starts or stops. With more than a hundred transition effects available, your streaming presentation can include special effects found in professional video production. You can also use transition effects to create a streaming slideshow from still images.

Understanding Transition Effects

A transition occurs every time a clip starts or stops playing. If you do not use a transition effect, the clip simply appears when it starts playing and, depending on its fill attribute, disappears when it stops playing. Using transition effects makes these transitions more visually compelling. Instead of just appearing onscreen, the clip might slowly fade in from a solid color. Or, a five-point star might expand from the center of a region to reveal the clip. Instead of just disappearing when it stops playing, a clip might crossfade into the clip that plays next.

Examples of Transition Effects

Timelines and Transition Effects

By default, each transition lasts one second, but you can make a transition last any length of time. Using transition effects does not affect a presentation's timeline. For example, a two-second transition applied to the end of a clip occurs during the last two seconds that the clip plays. If it is applied to the beginning of the clip, it occurs during the first two seconds of playback.

For More Information: For instructions on doing this, see "Setting a Transition Effect's Duration" on page 331.

Layouts and Transition Effects

You can use a transition effect with any visual clip regardless of the layout you've defined, or whether another clip precedes or follows the clip that uses the effect. When a clip starts, the area it covers is treated as its background, whether that area is a region color, a clip in another region, or a clip in the same region. A transition effect simply introduces the clip over, or removes the clip from, its background. So when you use transitions with a sequence of clips, the clips do not have to be the same size.

Animations and Transition Effects

Transition effects are distinct from the SMIL animations described in Chapter 16. A transition effect is a special effect that occurs when a clip starts or stops playing. An animation, on the other hand, is a special effect that occurs while a clip plays. You can use both transition effects and animations in the same presentation. You can even apply them to the same clip. But you define them separately.

Audio and Transition Effects

A transition effect does not change a clip's audio level. If you slowly fade into a video, for example, the audio plays normally throughout the fade. You can change a clip's audio level, however, by animating the clip region's `soundLevel` attribute. See "Controlling Audio Volume in a Region" on page 218 for information about `soundLevel`. Chapter 16 describes SMIL animations.

Multiple Clips with Transition Effects

Transition effects are applied to individual clips. Two clips playing in separate regions might end at the same time and use the same transition effect, such as a wipe transition that travels from left to right. In this case, two separate wipe transitions occur, one for each clip. Each transition effect is confined to the region in which the clip plays. You cannot make a single transition effect apply to both clips. For example, you cannot make the left-to-right wipe effect travel across the entire root-layout area, ushering in a new clip to each region as it passes over the region.

Summary of Transition Effects Tags

The following SMIL sample illustrates the functions and relationships of the tags used to create transition effects. The remainder of this chapter describes how to use these tags and their attributes to define and apply transition effects:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <head>
    <transition id="ID1" ...defines a transition type and duration.../>
    <transition id="ID2" ...defines a transition type and duration.../>
    ...
  </head>
  <body>
    <seq>
      <ref src="..." transIn="ID1" ...assigns a transition for the clip beginning.../>
      <ref src="..." transOut="ID2" ...assigns a transition for the clip end.../>
      ...
    </seq>
  </body>
</smil>
```

Defining Transition Types

The SMIL file header section defines the transition effects your presentation uses. The following example defines three transition effects after the layout:

```
<head>
  <layout>
    ...layout defined here...
  </layout>
```

```
  <transition id="fade1" type="fade" subtype="crossfade"/>
  <transition id="wipe1" type="pushWipe" subtype="fromTop"/>
  <transition id="rad1" type="radialWipe" subtype="counterTopBottom"/>
</head>
```

Each transition is defined by a separate <transition/> tag that typically has at least three attributes, which are described in the following table.

Basic Transition Effects Attributes

Attribute	Function
id	Sets a unique ID used to assign the transition to clips. For rules about creating IDs, see "SMIL Tag ID Values" on page 138.
type	Identifies a group of transition effects. This attribute is required.
subtype	Determines which member of the transition type group is used.

The following sections describe the various types and subtypes for transition effects. For convenience, the transition effects are grouped in families that share broad similarities, such as edge wipes and iris wipes. In defining a transition, you specify only the type and subtype, however.

> **Note:** Most transitions listed in the following sections have an SMPTE (Society of Motion Picture and Television Engineers) code. This code is provided for persons who want to find the SMIL transition effect that corresponds to a specific SMPTE transition. SMPTE codes are not used when defining SMIL transition effects, though.

> **Tip:** To display samples of transition effects in RealONE Player, get the HTML+Javascript version of this guide as described in "How to Download This Guide to Your Computer" on page 5.

Edge Wipe Transition Effects

In the edge wipe family, an "edge" moves over the first clip, revealing the second clip. As an analogy, imagine a car covered with snow. As the windshield wiper moves, its edge reveals the underlying windshield. In these transitions,

the edge may be different shapes, such as a straight line, a wedge, or a zigzag. The first subtype listed for each type in the following table is the default.

Edge Wipe Transition Effects

Type	Subtype	SMPTE	Transition Appearance
barWipe	leftToRight	1	A bar moves from left to right.
	topToBottom	2	A bar moves from top to bottom.
boxWipe	topLeft	3	A box expands from the upper-left corner to the lower-right corner.
	topRight	4	A box expands from the upper-right corner to the lower-left corner.
	bottomRight	5	A box expands from the lower-right corner to the upper-left corner.
	bottomLeft	6	A box expands from the lower-left corner to the upper-right corner.
	topCenter	23	A box expands from the top edge's midpoint to the bottom corners.
	rightCenter	24	A box expands from the right edge's midpoint to the left corners.
	bottomCenter	25	A box expands from the bottom edge's midpoint to the top corners.
	leftCenter	26	A box expands from the left edge's midpoint to the right corners.
fourBoxWipe	cornersIn	7	A box shape expands from each of the four corners toward the center.
	cornersOut	8	A box shape expands from the center of each quadrant toward the corners of each quadrant.
barnDoorWipe	vertical	21	A central, vertical line splits and expands toward the left and right edges.
	horizontal	22	A central, horizontal line splits and expands toward the top and bottom edges.
	diagonalBottomLeft	45	A diagonal line from the lower-left to upper-right corners splits and expands toward the opposite corners.
	diagonalTopLeft	46	A diagonal line from upper-left to lower-right corners splits and expands toward the opposite corners.

(Table Page 1 of 3)

Edge Wipe Transition Effects (continued)

Type	Subtype	SMPTE	Transition Appearance
diagonalWipe	topLeft	41	A diagonal line moves from the upper-left corner to the lower-right corner.
	topRight	42	A diagonal line moves from the upper right corner to the lower-left corner.
bowTieWipe	vertical	43	Two wedge shapes slide in from the top and bottom edges toward the center.
	horizontal	44	Two wedge shapes slide in from the left and right edges toward the center.
miscDiagonal Wipe	doubleBarnDoor	47	Four wedge shapes split from the center and retract toward the four edges.
	doubleDiamond	48	A diamond connecting the four edge midpoints simultaneously contracts toward the center and expands toward the edges.
veeWipe	down	61	A wedge shape moves from top to bottom.
	left	62	A wedge shape moves from right to left.
	up	63	A wedge shape moves from bottom to top.
	right	64	A wedge shape moves from left to right.
barnVeeWipe	down	65	A “V” shape extending from the bottom edge’s midpoint to the opposite corners contracts toward the center and expands toward the edges.
	left	66	A “V” shape extending from the left edge’s midpoint to the opposite corners contracts toward the center and expands toward the edges.
	up	67	A “V” shape extending from the top edge’s midpoint to the opposite corners contracts toward the center and expands toward the edges.
	right	68	A “V” shape extending from the right edge’s midpoint to the opposite corners contracts toward the center and expands toward the edges.

(Table Page 2 of 3)

Edge Wipe Transition Effects (continued)

Type	Subtype	SMPTE	Transition Appearance
zipZagWipe	leftToRight	71	A zigzag shape moves from left to right.
	topToBottom	72	A zigzag shape moves from top to bottom.
barnZigZag Wipe	vertical	73	The vertical, central line splits in a zigzag pattern and moves toward the left and right edges.
	horizontal	74	The horizontal, central line splits in a zigzag pattern and moves toward the top and bottom edges.

(Table Page 3 of 3)

Iris Wipe Transition Effects

A transition effect in the iris wipe family reveals a clip through an expanding shape. For example, a star can expand from the center of the transition area to reveal a new clip. The first subtype listed for each type in the following table is the default.

Iris Wipe Transition Effects

Type	Subtype	SMPTE	Transition Appearance
irisWipe	rectangle	101	A rectangle expands from the center.
	diamond	102	A four-sided diamond expands from the center.
triangleWipe	up	103	A triangle pointed toward the top edge expands from the center.
	right	104	A triangle pointed toward the right edge expands from the center.
	down	105	A triangle pointed toward the bottom edge expands from the center.
	left	106	A triangle pointed toward the left edge expands from the center.

(Table Page 1 of 2)

Iris Wipe Transition Effects (continued)

Type	Subtype	SMPTE	Transition Appearance
arrowHeadWipe	up	107	An arrowhead shape pointed toward the top edge expands from the center.
	right	108	An arrowhead shape pointed toward the right edge expands from the center.
	down	109	An arrowhead shape pointed toward the bottom edge expands from the center.
	left	110	An arrowhead shape pointed toward the left edge expands from the center.
pentagonWipe	up	111	A pentagon pointed toward the top edge expands from the center.
	down	112	A pentagon pointed toward the bottom edge expands from the center.
hexagonWipe	horizontal	113	A hexagon with flat sides at top and bottom expands from the center.
	vertical	114	A hexagon with flat sides at left and right expands from the center.
ellipseWipe	circle	119	A circle expands from the center.
	horizontal	120	A horizontal ellipse expands from the center.
	vertical	121	A vertical ellipse expands from the center.
eyeWipe	horizontal	122	An eye shape, its corners pointing left and right, expands from the center.
	vertical	123	An eye shape, its corners pointing up and down, expands from the center.
roundRectWipe	horizontal	124	A horizontal rectangle with rounded corners expands from the center.
	vertical	125	A vertical rectangle with rounded corners expands from the center.
starWipe	fourPoint	127	A four-pointed star expands from the center.
	fivePoint	128	A five-pointed star expands from the center.
	sixPoint	129	A six-pointed star expands from the center.
miscShapeWipe	heart	130	A heart shape expands from the center.
	keyhole	131	A keyhole shape expands from the center.

(Table Page 2 of 2)

Clock Wipe Transition Effects

The clock wipe family includes transition effects in which a clip is revealed by a radial sweep, similar to the second hand sweeping around the face of a clock. The first subtype listed for each type in the following table is the default.

Clock Wipe Transition Effects

Type	Subtype	SMPTE	Transition Appearance
clockWipe	clockwiseTwelve	201	A radial hand sweeps clockwise from the twelve o'clock position.
	clockwiseThree	202	A radial hand sweeps clockwise from the three o'clock position.
	clockwiseSix	203	A radial hand sweeps clockwise from the six o'clock position.
	clockwiseNine	204	A radial hand sweeps clockwise from the nine o'clock position.
pinWheelWipe	twoBladeVertical	205	Two radial hands sweep clockwise from the twelve and six o'clock positions.
	twoBladeHorizontal	206	Two radial hands sweep clockwise from the nine and three o'clock positions.
	fourBlade	207	Four radial hands sweep clockwise.
fanWipe	centerTop	211	A fan unfolds from the top edge, the fan axis at the center.
	centerRight	212	A fan unfolds from the right edge, the fan axis at the center.
	top	231	A fan unfolds from the bottom, the fan axis at the top edge's midpoint.
	right	232	A fan unfolds from the left, the fan axis at the right edge's midpoint.
	bottom	233	A fan unfolds from the top, the fan axis at the bottom edge's midpoint.
	left	234	A fan unfolds from the right, the fan axis at the left edge's midpoint.

(Table Page 1 of 4)

Clock Wipe Transition Effects (continued)

Type	Subtype	SMPTE	Transition Appearance
doubleFanWipe	fanOutVertical	213	Two fans, their axes at the center, unfold from the top and bottom.
	fanOutHorizontal	214	Two fans, their axes at the center, unfold from the left and right.
	fanInVertical	235	Two fans, their axes at the top and bottom, unfold from the center.
	fanInHorizontal	236	Two fans, their axes at the left and right, unfold from the center.
singleSweepWipe	clockwiseTop	221	A radial hand sweeps clockwise from the top edge's midpoint.
	clockwiseRight	222	A radial hand sweeps clockwise from the right edge's midpoint.
	clockwiseBottom	223	A radial hand sweeps clockwise from the bottom edge's midpoint.
	clockwiseLeft	224	A radial hand sweeps clockwise from the left edge's midpoint.
	clockwiseTopLeft	241	A radial hand sweeps clockwise from the upper-left corner.
	counterClockwise BottomLeft	242	A radial hand sweeps counter-clockwise from the lower-left corner.
	clockwiseBottom Right	243	A radial hand sweeps clockwise from the lower-right corner.
	counterClockwise TopRight	244	A radial hand sweeps counter-clockwise from the upper-right corner.

(Table Page 2 of 4)

Clock Wipe Transition Effects (continued)

Type	Subtype	SMPTE	Transition Appearance
doubleSweepWipe	parallelVertical	225	Two radial hands sweep clockwise and counter-clockwise from the top and bottom edges' midpoints.
	parallelDiagonal	226	Two radial hands sweep clockwise and counter-clockwise from the left and right edges' midpoints.
	oppositeVertical	227	Two radial hands attached at the top and bottom edges' midpoints sweep from right to left.
	oppositeHorizontal	228	Two radial hands attached at the left and right edges' midpoints sweep from top to bottom.
	parallelDiagonal TopLeft	245	Two radial hands attached at the upper-left and lower-right corners sweep down and up.
	parallelDiagonal BottomLeft	246	Two radial hands attached at the lower-left and upper-right corners sweep down and up.
saloonDoorWipe	top	251	Two radial hands attached at the upper-left and upper-right corners sweep down.
	left	252	Two radial hands attached at the upper-left and lower-left corners sweep to the right.
	bottom	253	Two radial hands attached at the lower-left and lower-right corners sweep up.
	right	254	Two radial hands attached at the upper-right and lower-right corners sweep to the left.

(Table Page 3 of 4)

Clock Wipe Transition Effects (continued)

Type	Subtype	SMPTE	Transition Appearance
windshieldWipe	right	261	Two radial hands attached at the midpoints of the top and bottom halves sweep from right to left.
	up	262	Two radial hands attached at the midpoints of the left and right halves sweep from top to bottom.
	vertical	263	Two sets of radial hands attached at the midpoints of the top and bottom halves sweep from top to bottom and bottom to top.
	horizontal	264	Two sets of radial hands attached at the midpoints of the left and right halves sweep from left to right and right to left.

(Table Page 4 of 4)

Matrix Wipe Transition Effects

The matrix wipe family includes transition effects in which a clip is revealed by a series of sequential tiles that follow a pattern, such as a spiral. In the following table, the first subtype listed for each type is the default.

Matrix Wipe Transition Effects

Type	Subtype	SMPTE	Transition Appearance
snakeWipe	topLeftHorizontal	301	Tiles move in a horizontal zigzag from the upper-left corner.
	topLeftVertical	302	Tiles move in a vertical zigzag from the upper-left corner.
	topLeftDiagonal	303	Tiles move in a diagonal zigzag from the upper-left corner.
	topRightDiagonal	304	Tiles move in a diagonal zigzag from the upper-right corner.
	bottomRightDiagonal	305	Tiles move in a diagonal zigzag from the lower-right corner.
	bottomLeftDiagonal	306	Tiles move in a diagonal zigzag from the lower-left corner.

(Table Page 1 of 4)

Matrix Wipe Transition Effects (continued)

Type	Subtype	SMPTE	Transition Appearance
spiralWipe	topLeftClockwise	310	Tiles spiral clockwise from the upper-left corner.
	topRightClockwise	311	Tiles spiral clockwise from the upper-right corner.
	bottomRightClockwise	312	Tiles spiral clockwise from the lower-right corner.
	bottomLeftClockwise	313	Tiles spiral clockwise from the lower-left corner.
	topLeft CounterClockwise	314	Tiles spiral counter-clockwise from the upper-left corner.
	topRight CounterClockwise	315	Tiles spiral counter-clockwise from the upper-right corner.
	bottomRight CounterClockwise	316	Tiles spiral counter-clockwise from the lower-right corner.
	bottomLeft CounterClockwise	317	Tiles spiral counter-clockwise from the lower-left corner.

(Table Page 2 of 4)

Matrix Wipe Transition Effects (continued)

Type	Subtype	SMPTE	Transition Appearance
parallelSnakes Wipe	verticalTopSame	320	Tiles move in two vertical zigzags, lines headed the same direction, starting from the upper-left and upper-right corners.
	verticalBottomSame	321	Tiles move in two vertical zigzags, lines headed the same direction, starting from the lower-left and lower-right corners.
	verticalTopLeft Opposite	322	Tiles move in two vertical zigzags, lines headed opposite directions, starting from the upper-left and lower-right corners.
	verticalBottomLeft Opposite	323	Tiles move in two vertical zigzags, lines headed opposite directions, starting from the lower-left and upper-right corners.
	horizontalLeftSame	324	Tiles move in two horizontal zigzags, lines headed the same direction, starting from the upper-left and lower-left corners.
	horizontalRightSame	325	Tiles move in two horizontal zigzags, lines headed the same direction, starting from the upper-right and lower-right corners.
	horizontalTopLeft Opposite	326	Tiles move in two horizontal zigzags, lines headed opposite directions, starting from the upper-left and lower-right corners.
	horizontalTopRight Opposite	327	Tiles move in two horizontal zigzags, lines headed opposite directions, starting from the upper-right and lower-left corners.
	diagonalBottomLeft Opposite	328	Two tile zigzags move outward in opposite directions from the diagonal line connecting the lower-left and upper-right corners.
	diagonalTopLeft Opposite	329	Two tile zigzags move outward in opposite directions from the diagonal line connecting the upper-left and lower-right corners.

(Table Page 3 of 4)

Matrix Wipe Transition Effects (continued)

Type	Subtype	SMPTE	Transition Appearance
boxSnakesWipe	twoBoxTop	340	Two lines of tiles spiral inward, starting in the upper corners and moving vertically.
	twoBoxBottom	341	Two lines of tiles spiral inward, starting in the lower corners and moving vertically.
	twoBoxLeft	342	Two lines of tiles spiral inward, starting in the left corners and moving horizontally.
	twoBoxRight	343	Two lines of tiles spiral inward, starting in the right corners and moving horizontally.
	fourBoxVertical	344	Four lines of tiles spiral inward, starting in the four corners and moving vertically.
	fourBoxHorizontal	345	Four lines of tiles spiral inward, starting in the four corners and moving horizontally.
waterfallWipe	verticalLeft	350	Tiles cascade vertically from the left in a waterfall effect.
	verticalRight	351	Tiles cascade vertically from the right in a waterfall effect.
	horizontalLeft	352	Tiles cascade horizontally from the left in a waterfall effect.
	horizontalRight	353	Tiles cascade horizontally from the right in a waterfall effect.

(Table Page 4 of 4)

Fade, Push, and Slide Transition Effects

This transition family, which has no corresponding SMPTE codes, includes fades that let you blend images into one another, or fade an image into or out of a solid color. The push and wipe transition effects allow a second clip to

push the first clip out of the way, or to slide over it. In the following table, the first subtype for a certain type is the default.

Fade, Push, and Slide Transition Effects

Type	Subtype	Transition Appearance
fade	crossfade	The clip fades into the clip that follows it.
	fadeFromColor	The clip fades in from a solid color.
	fadeToColor	The clip fades out into a solid color.
pushWipe	fromLeft	The clip pushes out the preceding clip from left to right.
	fromRight	The clip pushes out the preceding clip from right to left.
	fromTop	The clip pushes out the previous clip from top to bottom.
	fromBottom	The clip pushes out the previous clip from bottom to top.
slideWipe	fromLeft	The clip slides over the preceding clip from left to right.
	fromRight	The clip slides over the preceding clip from right to left.
	fromTop	The clip slides over the previous clip from top to bottom.
	fromBottom	The clip slides over the previous clip from bottom to top.

For More Information: With color fades, see See "Defining Colors and Border Blends" on page 334 for information on color values.

Modifying Transition Effects

The following sections describe optional <transition/> tag attributes that modify the appearance of the transition effects. The following table summarizes these attributes.

Attributes for Modifying Transition Effects

Attribute	Value	Default	Function	Reference
borderColor	blend|*color_value*	black	Defines the border color.	page 334
borderWidth	*pixels*	0	Specifies the border size.	page 334
direction	forward|reverse	forward	Sets the direction of movement.	page 331
dur	*time_value*	1s	Specifies effect duration.	page 331
endProgress	0.0-1.0	1.0	Halts the effect before it finishes.	page 332

(Table Page 1 of 2)

Attributes for Modifying Transition Effects (continued)

Attribute	Value	Default	Function	Reference
fadeColor	*color_value*	black	Sets color for fade transitions.	page 334
horzRepeat	*integer*	1	Multiplies the effect horizontally.	page 333
startProgress	0.0-1.0	0.0	Starts the effect at a midway point.	page 332
vertRepeat	*integer*	1	Multiplies the effect vertically.	page 333

(Table Page 2 of 2)

Setting a Transition Effect's Duration

By default, each transition effect lasts one second, but you can change this by adding a dur attribute to the <transition/> tag. As described in "Timelines and Transition Effects" on page 316, changing a transition effect's duration does not affect the presentation duration. In the following example, the transition effect takes three seconds to complete:

```
<transition id="fade1" type="fade" subtype="crossfade" dur="3s"/>
```

To use the same transition type but vary the transition speeds, define the transition multiple times, each time with a different ID and duration. For example, the following tags define the same transition type and subtype, but the first effect lasts two seconds whereas the second effect lasts four seconds:

```
<transition id="fan1" type="fanWipe" subtype="top" dur="2s"/>
<transition id="fan2" type="fanWipe" subtype="top" dur="4s"/>
```

> **For More Information:** The dur attribute uses the standard SMIL timing values, which are described in "Specifying Time Values" on page 239.

Reversing a Transition Effect's Direction

Using direction="reverse", you can change the direction a transition effect runs. For example, the following transition effect reveals the clip in a four-point star that expands outward:

```
<transition id="p1" type="starWipe" subtype="fourPoint"/>
```

Reversing the direction creates a four-point star that contracts inward:

```
<transition id="p2" type="starWipe" subtype="fourPoint" direction="reverse"/>
```

For some transition effects, you can simply use a different subtype rather than include the attribute direction="reverse". For example, this transition effect:

```
<transition id="p3" type="pushWipe" subtype="fromRight"/>
```

is equivalent to this transition effect:

```
<transition id="p4" type="pushWipe" subtype="fromLeft" direction="reverse"/>
```

Note: Reversing the direction of a transition effect that has no specific starting or ending point, such as crossfade, has no visual effect.

Using Partial Transition Effects

Each transition effect has a starting appearance and an ending appearance. For example, an expanding star transition normally starts as a single point in the center of the transition area. It ends after the star has expanded out of the transition area. You can set a different point where a transition effect starts with startProgress:

```
<transition id="wipe1" type="pushWipe" subtype="fromLeft" startProgress="0.25"/>
```

The startProgress attribute takes a value from 0.0 (normal starting point) to 1.0 (normal ending point). This value represents a percentage. For example, startProgress="0.25" means that when the transition effect starts, it appears to be 25 percent complete already. It then flows to its end point over the course of its specified duration.

Additionally, you can use endProgress, which also takes a value from 0.0 to 1.0 to indicate how far the transition effect progresses before it ends. The following example defines a keyhole-shape transition effect that ends when the keyhole has expanded to half of its normal ending size:

```
<transition id="key" type="miscShapeWipe" subtype="keyhole" endProgress="0.5"/>
```

Tips for Using Partial Transition Effects

- When you use endProgress, the transition effect ends in an intermediate state. You can use this to create special effects with iris wipes, for example. With other types of transition effects, though, a partially completed transition may confuse the viewer.
- You can combine the startProgress and endProgress attributes in a single <transition/> tag. When you do this, the endProgress value must be equal to, or higher than, the startProgress value for the transition effect to exhibit any movement.

- If you set the `startProgress` and `endProgress` attributes to the same value in a `<transition/>` tag, the transition effect appears to complete instantly, regardless of its duration.
- When you use a partial transition effect to introduce a new clip in a sequence, the preceding clip's `fill` attribute determines whether parts of that clip remain visible at the end of the effect. Use one of the following:
  - `fill="hold"` to keep the first clip visible
  - `fill="transition"` to make the first clip disappear after the transition completes
  - `fill="remove"` to make the first clip disappear before the transition begins

For More Information: See "Using Clip Fills with Transition Effects" on page 336 for more information.

Repeating Transition Effects Horizontally or Vertically

When you repeat a transition effect, the effect appears multiple times instead of just once. For example, an expanding star transition effect normally begins in the center of the clip and expands toward the clip's edges. By repeating this effect twice horizontally and twice vertically, you make a separate star shape expand in each of the clip's quadrants, as shown in the following illustration.

Repeating Star Transition Effect

You repeat a transition effect by adding the `horzRepeat` or `vertRepeat` attribute to a `<transition/>` tag. Each attribute takes as a value a positive integer that defines how many times the transition effect repeats horizontally or vertically,

respectively. For example, the following transition effect defines two four-point stars that appear side-by-side:

```
<transition id="starHorz" type="starWipe" subtype="fourPoint" horzRepeat="2"/>
```

To have these stars appear one on top of the other, you repeat the effect vertically:

```
<transition id="starVert" type="starWipe" subtype="fourPoint" vertRepeat="2"/>
```

You can combine horzRepeat and vertRepeat attributes in the same tag. The following example creates a grid of nine transition effects by defining three horizontal repetitions and three vertical repetitions:

```
<transition id="nineStar" type="starWipe" subtype="fourPoint" horzRepeat="3"
 vertRepeat="3"/>
```

> **Tip:** Think of these attributes as defining a table. The horzRepeat attribute defines the number of columns, and the vertRepeat attribute defines the number of rows.

Setting a Border Width

All transition effects except fades have borders. When a clip slides over another clip from left to right, for example, the border is the new clip's right edge. By default, the border width is 0 (zero), meaning the border is not accentuated. By adding a borderWidth attribute to a <transition/> tag, you can make the border more apparent. This attribute takes as a value a positive integer that sets the border's pixel width. The following example sets a two-pixel border width:

```
<transition id="wipe1" type="pushWipe" subtype="fromLeft" borderWidth="2"/>
```

By default, the border is black, but you can use any other RGB color. You can also make the border blend the clip with its background. The following section explains how to do this.

Defining Colors and Border Blends

Transition effects that fade to or from a color, as well as transition effects that set border widths, can include color values, which are described in Appendix D. The following example defines a transition effect in which the clip fades to a solid red:

```
<transition id="redFade" type="fade" subtype="fadeToColor" fadeColor="red"/>
```

If you define a border width as described in the preceding section, you can use the borderColor attribute to set the border color:

```
<transition id="wipe1" type="pushWipe" subtype="fromLeft" borderWidth="2"
 borderColor="#AFBC08"/>
```

Alternatively, you can use borderColor="blend" to make the border blend the clip into its background. This typically creates a blurring effect along the border:

```
<transition id="wipe1" type="pushWipe" subtype="fromLeft" borderWidth="16"
 borderColor="blend"/>
```

> **Tip:** When using borderColor="blend", you typically need to set borderWidth to 10 pixels or higher to notice the blending effect.

Assigning Transition Effects to Clips

After you define <transition/> tags in the SMIL file header, you assign the transition effects to clips using transIn and transOut attributes in each clip source tag. You can assign transition effects only to clip source tags, not to <seq>, <par>, or <excl> groups. Any type of clip can use a transition, but because transitions are visual, they have no effect on a clip's audio track.

The transIn attribute makes the transition effect occur as the clip starts to play. The transOut attribute makes the effect occur as the clip finishes playing. Each attribute takes as a value the ID defined in a <transition/> tag. For example, suppose that you define the following two transition effects:

```
<transition id="fromBlue" type="fade" subtype="fadeFromColor" fadeColor="blue"/>
<transition id="toBlue" type="fade" subtype="fadeToColor" fadeColor="blue"/>
```

In the SMIL file body, you could the assign the effects to a sequence of two videos like this:

```
<seq>
  <video src="video1.rm" transIn="fromBlue" transOut="toBlue" .../>
  <video src="video2.rm" transIn="fromBlue" transOut="toBlue" .../>
</seq>
```

In the preceding example, each video fades up from a solid blue when it starts, then fades down to solid blue when it ends. It's not necessary to use both the transIn and transOut attributes for each clip, though. In the following example, the first video starts playing without any transition. As the first clip ends and the second clip starts, there's a fade to blue and then a fade up. When the second clip stops playback, it simply disappears from the screen:

```
<seq>
  <video src="file://video1.rm" transOut="toBlue"/>
  <video src="file://video2.rm" transIn="fromBlue" fill="remove"/>
</seq>
```

> **Note:** A transition effect assigned with a transOut attribute always obeys SMIL timing rules. If a video normally plays for two minutes, but has a dur="3min" value to lengthen its active period, the transition effect occurs after three minutes.

Using Clip Fills with Transition Effects

The section "Setting a Fill" on page 251 explains the fill attribute, which makes the clip disappear or remain visible when it finishes playing. Which values you use for fill can also affect transition effects. The following sections explain how best to use the fill attribute with transition effects.

Defining a Transition Fill for a Sequence of Clips

When you apply transition effects to a sequence of clips, use fill="transition" to keep a clip onscreen long enough for a transition to occur. The transition value does nothing when a transition is not applied to the clip. Suppose you want to use a three-second radial wipe like the following to introduce each new video in a sequence:

```
<transition id="fan1" type="fanWipe" subtype="top" dur="3s"/>
```

You could apply this transition effect to the beginning of each clip. In a standard sequence of clips, though, each clip disappears as soon as it stops playing. The transition effect that introduces the next clip therefore operates against the region's background color. To keep clips onscreen during transitions, add fill="transition" to each clip's source tag:

```
<seq>
  <video src="video1.rm" transIn="fan1" fill="transition"/>
  <video src="video2.rm" transIn="fan1" fill="transition"/>
  ...more clips that use fill="transition"...
  <video src="video6.rm" transIn="fan1" fill="remove"/>
</seq>
```

In this sequence, each fill="transition" attribute keeps the clip onscreen for three seconds (the duration of the transition effect) after the clip ends playback, long enough for the transition effect to complete. This does not

lengthen the presentation timeline. The three seconds used for each transition effect overlap the first three seconds that each new clip plays.

> **Note:** If a clip in a sequence uses a `begin` value to delay its playback, a `fill="transition"` value in the preceding clip freezes that clip until the clip with the `begin` value starts to play and the transition effect completes. For more on `begin`, see "Using a Begin Time with a Clip" on page 240.

> **Tip:** In a long sequence of clips, add `fillDefault="transition"` to the `<seq>` tag. You do not then need to add `fill="transition"` to every clip tag. For more on `fillDefault`, see "Specifying a Default Fill" on page 258.

Setting a Fill in Parallel Groups

When you use parallel groups, a `fill="transition"`, `fill="remove"` or `fill="freeze"` attribute in a clip source tag can affect when a transition occurs. Suppose that you define a two-second fade to black:

```
<transition id="toBlack" type="fade" subType="fadeToColor" dur="2s"/>
```

You next apply this transition to both an image and a video playing in parallel. In the following example, the image clip has a `fill="remove"` attribute and a 30-second duration. The clip begins to fade out at 28 seconds into the parallel group's timeline, disappearing much sooner than the video, which has a 154-second duration:

```
<par>
  <img src="..." region="image" transOut="toBlack" fill="remove" dur="30s"/>
  <video src="..." region="video" transOut="toBlack" dur="154s"/>
</par>
```

To make the image fade out only after its duration has elapsed, you would use `fill="transition"` as shown in the following example. In this case, the image disappears 32 seconds after it begins to play:

```
<par>
  <img src="..." region="image" transOut="toBlack" fill="transition" dur="30s"/>
  <video src="..." region="video" transOut="toBlack" dur="154s"/>
</par>
```

To make the image begin to fade out two seconds before the video finishes playing, you would use `fill="freeze"` as shown in the following example:

```
<par>
  <img src="..." region="image" transOut="toBlack" fill="freeze" dur="30s"/>
  <video src="..." region="video" transOut="toBlack" dur="154s"/>
</par>
```

Transition Effects Examples

The following sections illustrate how to use transition tags and attributes to create various transition effects. To see more examples, get the HTML+Javascript version of this guide as described in "How to Download This Guide to Your Computer" on page 5, and view the **Sample Files** page.

Fading to a Color Between Clips

One of the simplest transition effects is to fade up from or down to a color. The following example shows a sequence of two videos. There is a two-second fade from blue as each video starts, and a two-second fade to blue when each video ends. Each video is centered within the video region and appears at its normal size. A `begin="2s"` value is used with each video to insert a short delay before each transition occurs:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <head>
    <layout>
      <root-layout width="320" height="240" backgroundColor="blue"/>
      <region id="video_region"/>
      <regPoint id="middle" left="50%" top="50%" regAlign="center"/>
    </layout>
    <transition id="fromBlue" type="fade" subtype="fadeFromColor"
      fadeColor="blue" dur="2s"/>
    <transition id="toBlue" type="fade" subtype="fadeToColor"
      fadeColor="blue" dur="2s"/>
  </head>
  <body>
    <seq>
      <video src="..." region="video_region" regPoint="middle"
        transIn="fromBlue" transOut="toBlue" begin="2s" fill="remove"/>
      <video src="..." region="video_region" regPoint="middle"
        transIn="fromBlue" transOut="toBlue" begin="2s" fill="remove"/>
    </seq>
  </body>
</smil>
```

Crossfading Videos

In a simple variation of the preceding example, the first video fades up from blue when it starts and the second video fades down to blue when it ends. When the first video stops and the second video starts, however, the two videos crossfade into each other. Because crossfades look better when both clips are the same size, the clips are played in the same region, which uses fit="fill" to expand or shrink the clips to fill the region fully:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
 <head>
  <layout>
   <root-layout width="250" height="180" backgroundColor="blue"/>
   <region id="video_region" fit="fill"/>
  </layout>
  <transition id="fromBlue" type="fade" subtype="fadeFromColor"
   fadeColor="blue" dur="2s"/>
  <transition id="toBlue" type="fade" subtype="fadeToColor"
   fadeColor="blue" dur="2s"/>
  <transition id="xFade" type="fade" subtype="crossfade" dur="2s"/>
 </head>
 <body>
  <seq>
   <video src="..." region="video_region" transIn="fromBlue"
    begin="2s" fill="transition"/>
   <video src=".." region="video_region" transIn="xFade"
    transOut="toBlue" fill="remove"/>
  </seq>
 </body>
</smil>
```

CHAPTER

16

ANIMATIONS

Using SMIL animations, you can transform clips by expanding them, for example, or moving them around the screen. To use this advanced SMIL feature, you must thoroughly understand clip tags, groups, timing, and layouts as described in the preceding chapters. For information on Flash animation rather than SMIL animation, see Chapter 5.

> **Tip:** To see animation examples, get the HTML+Javascript version of this guide as described in "How to Download This Guide to Your Computer" on page 5, and view the **Sample Files** page.

Understanding Animations

SMIL animations provide the means for manipulating clips playing in RealONE Player. They are not themselves distinct clips. Instead, they are SMIL tags and attributes that instruct RealONE Player to modify a clip, whether a video, a still image, a brush object, or any other type of clip. You can even apply a SMIL animation to a Flash animation clip to "animate an animation." Common uses of SMIL animation include:

- expanding or shrinking a clip,
- moving a clip around the screen,
- changing a region's background color,
- boosting or cutting a clip's sound level, and
- altering a clip's transparency to make it more, or less, opaque.

> **Tip:** Chapter 15 explains transition effects, which are special effects that occur when a clip starts or stops playing. You can use transition effects and animations in the same presentation.

You can even apply them to the same clip. But you define them separately.

Animation Tags

You can add an animation to your SMIL presentation using any one of four animation tags:

- <animate/>

  The <animate/> tag is the principal tag used to create animations. The other tags are variations of the <animate/> tag, so once you learn how to use <animate/>, you will master the other tags quickly. The section "Creating Basic Animations" on page 345 explains the main attributes and values of the <animate/> tag.

- <animateColor/>

  The <animateColor/> tag is a variation of the <animate/> tag that works for color animations only. See the section "Animating Colors" on page 358 for more on this tag.

- <animateMotion/>

  The <animateMotion/> tag lets you move a clip both horizontally and vertically at the same time. A single <animate/> tag creates motion in only one direction. Thus, a single <animateMotion/> tag can do the work of two <animate/> tags. The section "Creating Horizontal and Vertical Motion" on page 359 explains how to use this tag.

- <set/>

  The <set/> tag instantly sets an animation. With an <animate/> tag, you can widen a region over the course of several seconds, for example. With the <set/> tag, in contrast, you can set the new width instantly. The section "Setting an Attribute Value" on page 360 describes the <set/> tag.

> **Tip:** Do not confuse the SMIL animation tags with the <animation/> tag, which is a clip source tag that introduces an animation clip into a presentation. For more on <animation/>, see "Creating Clip Source Tags" on page 145.

Animation Tag Placement

Animation tags always appear in the SMIL <body> section, even when they modify elements defined in the SMIL header, such as <region/> tags. Animation tags function much like clip source tags. You can place them in

groups, but you can also include them within clip source tags. The following sections describe the various means of adding animation tags to a SMIL file.

In a Clip Source Tag

When you want to animate a clip as it plays, you can turn the clip source tag into a binary tag, as shown in the following example:

```
<video ...>
  <animate ...animation for the video clip or region.../>
</video>
```

In this case, the animation typically affects the clip or the region playing the clip. Non-interactive timing attributes in the animation tag are relative to the start of clip playback. For example, a `begin="5s"` attribute in the animation tag starts the animation five seconds after the clip begins to play.

> **For More Information:** See "Binary and Unary Tags" on page 137 for the basics of modifying a clip source tag to include other SMIL elements.

In a Parallel Group

Because animations function like clip source tags, you can place them in parallel groups with other clips, as shown in the next example:

```
<par>
  <video.../>
  <textstream.../>
  <animate ...animation for any SMIL element.../>
</par>
```

In this case, the animation might apply to a clip in the same parallel group, or to any other element in the file. The animation plays only while its `<par>` group is active, however, and non-interactive timing attributes in the animation tag are relative to the start of the `<par>` group. For example, a `begin="10s"` attribute in the animation tag starts the animation 10 seconds after the group becomes active.

In a Sequence

Although not as common as the preceding cases, an animation can also be part of a sequence as shown here:

```
<seq>
  <video ... fill="hold"/>
  <animate ...animation for the preceding clip.../>
</seq>
```

Although a sequential animation can affect any SMIL element, it typically targets the preceding clip. Because the animation plays only when the preceding clip finishes, that clip typically uses `fill="hold"` to keep it from disappearing when it ends playback. Non-interactive timing attributes in the animation tag are relative to the end of the preceding clip. For example, a `begin="3s"` attribute in the animation tag starts the animation three seconds after the preceding clip finishes.

SMIL Timing with Animations

Because animations function like clip source tags, you can use SMIL timing attributes to control when animations start, and how long they last. The following are the most common timing attributes used with animations:

- `begin`

  The `begin` attribute, which is described in "Setting Begin and End Times" on page 240, controls when the animation starts, relative to the group or clip that contains the animation. If you do not use a `begin` value, the animation starts as soon as the clip or group that contains it becomes active. You can also use advanced `begin` times as described in Chapter 13 to start an animation when the screen pointer moves over a clip, for example.

- `dur` or `end`

  The `dur` or `end` attribute controls how long the animation lasts. As with any SMIL element, the `end` attribute works with `begin` to set a total playback time. For more information, see "Setting Durations" on page 243, as well as "Choosing end or dur" on page 243.

- `fill`

  The effects of an animation reset as soon as the animation's duration elapses unless you use a `fill` attribute. If the animation is in a `<par>` group and you use `fill="freeze"`, for example, the animation holds its final appearance until the group ends. The `fill` attribute is described in "Setting a Fill" on page 251.

- repeatCount or repeatDur

  You can make an animation replay several times with repeatCount or repeatDur. A repeating animation can also grow with each iteration. When widening a region, for example, you can make the region increase a certain amount on each repetition. The attributes for making animations repeat are described in "Repeating an Element" on page 247.

Simultaneous Animations

Several animations can occur at the same time during a presentation, as long as they do not conflict. You cannot increase and decrease a region's width at the same time, for example. But you can decrease its width, increase its height, move its left offset, and change its background color simultaneously by using several <animate/> tags that are active at the same time.

Creating Basic Animations

The <animate/> tag is the most versatile animation tag. You can use it to alter element sizes, positions, colors, and sound levels. The following table lists the attributes that you use to define animations with the <animate/> tag. Keep in mind, too, that animation tags typically use SMIL timing attributes, as described in "SMIL Timing with Animations" on page 344.

<animate/> Tag Attributes

Attribute	Value	Function	Reference
accumulate	none|sum	Makes a repeating animation build with each iteration.	page 356
additive	replace|sum	Adds the animation value to the existing value.	page 355
attributeName	*attribute_name*	Selects the attribute to animate.	page 346
by	*pixels*|*percentage*|*color_value*	Animates the element by a certain amount. Do not use with to.	page 351
calcMode	discrete|linear|paced	Controls the flow of an animation.	page 353
from	*pixels*|*percentage*|*color_value*	Sets a starting point for the animation. Use with to or by.	page 350
targetElement	*ID*	Identifies the tag that contains the animated attribute.	page 346

(Table Page 1 of 2)

<animate/> Tag Attributes (continued)

Attribute	Value	Function	Reference
to	*pixels*|*percentage*|*color_value*	Sets an end point for the animation. Do not use with by.	page 351
values	*pixels*|*percentage*|*color_value*	Defines a list of animation values. Not used with from, to, or by.	page 352

(Table Page 2 of 2)

Selecting the Element and Attribute to Animate

Using the targetElement attribute, you specify the ID of the SMIL element you want to animate. Using attributeName you select a specific attribute within that element. To animate a region's width, for example, you identify the region and its width attribute through an <animate/> tag in the SMIL body, as shown in the following example:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
 <head>
  ...
   <region id="video_region" width="320" height="240"/>
  ...
 </head>
 <body>
  ...
  <animate targetElement="video_region" attributeName="width" .../>
 </body>
</smil>
```

When the animation is within a clip source tag, attributeName is required, but targetElement is not necessary. In the following example, the <animate/> tag falls within the clip source tag. The <animate/> tag does not therefore need a targetElement attribute to select the video clip for animation:

```
<video ...>
 <animate attributeName="..." .../>
</video>
```

Animating Window Attributes

The following table describes the attributes that you can animate in <root-layout/> and <topLayout> tags. That is, you can use any of the following as values for attributeName when targetElement identifies an ID in a <root-layout/>

or <topLayout> tag. By animating these attributes, you can change the window size or alter its color.

<root-layout/> and <topLayout> Attribute Values You Can Animate

Attribute	Effect	Reference
backgroundColor	Modifies the window's background color. You could change the window's background color from black to white midway through a presentation, for example.	page 215
height	Modifies the window height. You can animate this attribute along with width to change the presentation's display size.	page 203
width	Alters the window's width. You can animate this attribute along with height to change the presentation's display size.	page 203

Animating Region Attributes

The next table lists all the attributes that you can animate in <region/> tags. In other words, you can use any of the following as values for attributeName when targetElement identifies a region ID. By animating these attributes, you can change a region's size, move the region around a window, alter its color, or change the volume of a playing clip.

Region Attribute Values You Can Animate

Attribute	Effect	Reference
backgroundColor	Modifies the region's background color. You could change the region's background color from black to white midway through a presentation, for example.	page 215
bottom	Changes the region's bottom offset. Animating this attribute can make the region taller or shorter, as well as move it vertically.	page 208
height	Modifies the region height. You can animate this attribute along with width to change a clip's size.	page 208
left	Changes the region's left offset. Animating this attribute can change the region's width, or move the region horizontally.	page 208
regionName	Moves a clip from region to region.	page 207
right	Changes the region's right offset. Animating this attribute can make the region wider or narrower, as well as move it horizontally.	page 208

(Table Page 1 of 2)

Region Attribute Values You Can Animate (continued)

Attribute	Effect	Reference
soundLevel	Adjusts a clip's sound level. You can animate this attribute to fade the clip's audio in or out.	page 218
top	Changes the region's top offset. Animating this attribute can make the region taller or shorter, as well as move it vertically.	page 208
width	Alters the region's width. You can animate this attribute along with height to change a clip's size.	page 208
z-index	Changes the region's stacking order. You can animate this attribute to bring one region in front of another region.	page 214

(Table Page 2 of 2)

Tips for Animating Regions

- An attribute does not have to be explicitly declared to be animated. You can animate the region's right attribute, for instance, even if that attribute is not defined in the <region/> tag.
- By animating the left, right, top, or bottom attributes, you can move the region around the <root-layout/> or the <topLayout> area. You can even move part or all of the region out of the display window. The region and any clips displaying in it are truncated at the window borders, however.
- If the animated region is a subregion (a region contained within another region), it will not display outside of the containing region. So if you move the subregion outside of its containing region, the subregion is truncated at the containing region's borders.
- If you move a region over another region, the regions' z-index values determine which region appears in front. You can also animate the z-index values to change this stacking order.

  Note: If region A appears on top of region B, you cannot animate a subregion in region B so that it appears on top of region A. For more information on subregion z-index values, see page 219.

- A region's fit attribute affects how a clip displays as a region's height or width changes. For more on fit, see "Defining How Clips Fit Regions" on page 227.

- When you animate the size of a clip that includes a hot spot hyperlink, the link expands or contracts with the clip if the hyperlink has no `coords` values, or its coordinates are defined with percentage values. Although the link does not change size if it is defined with pixel values, it is truncated if the region boundaries overlap the hot spot boundaries.

  For More Information: See "Defining Hot Spots" on page 285 for more on creating hot spots.

- You cannot animate attributes in `<regPoint>` or `<transition>` tags.

Animating Clip Attributes

The following table lists all the attributes that you can animate in clip source tags. That is, you can use any of the following as values for `attributeName` when `targetElement` identifies a tag such as `<video/>`, `<img/>`, or `<ref/>`.

Clip Attribute Values You Can Animate

Attribute	Effect	Reference
`backgroundColor`	Modifies the background color of the region playing the clip. You could change the color from red to blue midway through a presentation, for example.	page 220
`rn:backgroundOpacity`	Modifies the opacity in a clip's background transparency. You could make the clip more opaque, for example.	page 158
`bottom`	Changes the clip's bottom offset from its playback region. Animating this attribute can make the clip taller or shorter, as well as move it vertically.	page 220
`color`	Changes the color of a `<brush/>` object.	page 148
`height`	Modifies the clip's height. You can animate this attribute along with `width` to change a clip's size.	page 220
`left`	Changes the clip's left offset. Animating this attribute can make the clip narrower or wider, as well as move it horizontally.	page 220
`rn:mediaOpacity`	Turns opaque areas in the clip transparent. By animating this attribute, you can make the clip blend in with the region background color.	page 158
`right`	Changes the clip's right offset. Animating this attribute can make the clip narrower or wider, as well as move it horizontally.	page 220

(Table Page 1 of 2)

Clip Attribute Values You Can Animate (continued)

Attribute	Effect	Reference
top	Changes the clip's top offset. Animating this attribute can make the clip taller or shorter, as well as move it vertically.	page 220
width	Alters the clip's width. You can animate this attribute along with height to change a clip's size.	page 220
z-index	Changes the clip's stacking order. You can animate this attribute to bring one clip in front of another.	page 220

(Table Page 2 of 2)

Tips for Animating Clip Source Tags

- An attribute does not have to be explicitly declared. You can animate a width attribute, for example, even if it is not explicitly defined in a clip source tag
- Animating a size or position attribute (such as width or top) in a clip creates or modifies a single-use subregion that holds the clip. Therefore, the points about animated regions described in "Tips for Animating Regions" on page 348 apply to these types of clip animations.

  For More Information: The section "Defining Single-Use Subregions" on page 220 explains these types of subregions.

- An animated clip cannot display outside of its playback region. To move a clip anywhere within the <root-layout/> or <topLayout> area, animate the main region that contains the clip, rather than the clip itself.

Animating Hot Spot Attributes

An animation tag can select the coords attribute of an <area/> tag to change the shape of a hot spot hypertext link. For information on the <area/> tag and the coords attribute, see "Using the <area/> Tag" on page 283.

Defining Simple Animation Values

Three animation attributes, to, by, and from, provide a simple means of defining where an animation starts and stops. Use either the to or the by attribute, but not both, to determine the animation end point. With either of these attributes, you can use the optional from attribute to change the animation's starting point.

Animating an Attribute to a Certain Point

The to attribute defines the animation's end point. It takes a value of the type appropriate for the animated attribute. When animating a layout attribute, for example, use a pixel or percentage value, either positive or negative. When animating a color, use a color name or value. For example, suppose that you have defined this region:

```
<region id="video_region" width="320" height="240" backgroundColor="green"/>
```

You could change the background color to yellow over the course of five seconds with an animation tag like the following:

```
<animate targetElement="video_region" attributeName="backgroundColor" to="red"
dur="5s"/>
```

Using the to attribute, you could also animate the region's size or placement. When it becomes active, the following <animate/> tag expands the region's width to 380 pixels over the course of three seconds:

```
<animate targetElement="video_region" attributeName="width" to="380" dur="3s"/>
```

If you do not include a from attribute, the animation starts at the value specified in the target element tag. In the preceding example, the animation starts at the region's normal width of 320 pixels. If you specify a from value, though, the region expands or contracts to that size instantly when the animation becomes active. With the following animation, the region first contracts to half its defined size, then grows to 380 pixels over five seconds:

```
<animate targetElement="video_region" attributeName="width" from="160" to="380"
dur="5s"/>
```

Animating an Attribute by a Certain Value

The by attribute defines a certain value by which the animation progresses. Use it to animate sizes or positions, but not colors. The by attribute can take a pixel or a percentage value, either positive or negative. Suppose that you want to expand the width of the following region:

```
<region id="video_region" width="320" height="240"/>
```

When it becomes active, the following <animate/> tag expands the region's width by 30 pixels to a final width of 350 pixels:

```
<animate targetElement="video_region" attributeName="width" by="30" dur="3s"/>
```

Because no from value is specified, the animation starts with the region's defined width. If you specified a different from value, the region would expand or contract to that size instantly when the animation became active.

Tips for Defining Simple Animation Values

- By default, an animation flows smoothly over the course of its duration. But you can use `calcMode="discrete"` to make the animation jump from its starting point to its stopping point. See "Controlling How an Animation Flows" on page 353 for more information.
- When animating a layout attribute, you can use a pixel value for the `to` or `by` attribute even if the region or subregion is defined with percentages, and vice versa.
- You can use negative pixel or percentage values when animating a size or a position. For example, animating a region width with `by="-25%"` shrinks the region to three-quarters of its normal size. Specifying `by="-40"` for a region's `left` attribute moves the region 40 pixels to the left, whereas using `by="40"` moves the region 40 pixels to the right.
- Not all negative values are useful. For example, using a negative value with the `to` attribute when animating a size (such as `to="-25%"` or `to="-44"` when animating a region width) causes the element to disappear completely.

Defining a Range of Animation Values

Using a values list, you can animate an attribute through multiple values. This lets you define animations that are more complex than those possible with the `to`, `by`, and `from` attributes. In the following example, a `values` attribute animates a region's width to four different sizes over 15 seconds:

```
<animate targetElement="video_region" attributeName="width" dur="15s"
values="58;150;96;110"/>
```

Tips for Defining a Values List

- Enclose the entire values list in double quotation marks.
- You can include spaces before or after a semicolon that separates values, but spaces are not necessary.
- You do not need to add a semicolon after the last value.
- As with the `to` and `by` attributes, you can specify negative pixel or percentage values when animating sizes and placements.
- The animation always proceeds in order from the first value to the last value. The first value is applied when the animation activates. The animation reaches the last value by the end of its duration.

- In a values list, use values appropriate to the animated attribute. When animating a region, for example, use percentages, pixels, or a mix of both:

  `values="25%;50%;380"`

- You can use a values list to animate colors. List either color names or color codes, as in `values="blue;green;#3FD233;rgb(255,12,192)"`.
- A list with only two values is equivalent to using the `from` and `to` attributes. For example, `values="58;150"` functions the same as `from="58"` and `to="150"`.

Controlling How an Animation Flows

The `calcMode` attribute, which works with the `values`, `to`, and `by` attributes, controls how the animation flows from point to point. It has three possible values, as described in the following table.

calcMode Attribute Values

Value	Function	Reference
`discrete`	Makes the animated element jump from value to value.	page 353
`linear`	Causes the animated element to flow smoothly from value to value, with the movement from each value taking an equal amount of time. This is the default value for `<animate/>` and `<animateColor/>`.	page 354
`paced`	Makes the animated element flow smoothly from value to value, with the movement evenly paced throughout the entire animation. With `to` and `by`, this functions the same as `linear`. This is the default value for `<animateMotion/>`.	page 354

Jumping from Value to Value

The `discrete` value for `calcMode` causes the animation to jump from point to point in the `values` list. For example, the following tag animates a region's width to four values over the course of eight seconds:

```
<animate targetElement="video_region" attributeName="width" dur="8s"
values="50;75;150;100" calcMode="discrete"/>
```

When the animation begins, the region's width is set to 50 pixels. At two seconds, the width jumps up to 75 pixels. At four seconds, it jumps up to 150 pixels. And at six seconds, it jumps down to 100 pixels, staying at that size for the remaining two seconds of the duration. Note that the last value is reached

at six seconds, rather than at the end of the eight-second duration. This makes each value active for an equal stretch (2 seconds) of the 8-second duration.

The calcMode="discrete" value also works with the to and by attributes to make the animation jump to its ending point. In the following example, the region width stays at 160 pixels through the first three seconds of the animation, then jumps to 320 pixels for the last three seconds:

```
<animate targetElement="video_region" attributeName="width" from="160" to="320"
calcMode="discrete" dur="6s"/
```

Moving Linearly from Point to Point

The following animation uses the default value calcMode="linear" to animate a region's width between four points over the course of 9 seconds:

```
<animate targetElement="video_region" attributeName="width" dur="9s"
values="50;75;200;100" calcMode="linear"/>
```

When the animation begins, the region's width is set to 50 pixels. It then begins to expand, reaching 75 pixels at three seconds. By six seconds, the width has grown to 200 pixels. It then begins to contract, reaching 100 pixels at nine seconds. So in contrast to the calcMode="discrete" example in the preceding section, the last value in this calcMode="linear" example is reached at the very end of the animation duration.

The speed of each segment in this animation increases with the distance between points. Notice that the distance from the first to the second point is 25 pixels, whereas the distance from the second to the third point is 125 pixels. Because each point-to-point expansion or contraction takes an equal amount of time, the speed of movement from the first to the second point is slower than the speed of movement from the second to the third point.

Flowing at an Even Pace

If you use calcMode="paced", movement flows smoothly over the course of the entire animation. When the following animation starts, the region width is 50 pixels. The region then expands at an even pace to 150 pixels before contracting at the same pace to 100 pixels:

```
<animate targetElement="video_region" attributeName="width" dur="6s"
values="50;150;100" calcMode="paced"/>
```

Because the animation is paced, the distance between points in the values list affects how long each phase takes. The distance of the expansion phase is 100

pixels (150 - 50), whereas the distance of the contraction phase is 50 pixels (150 - 100). The expansion therefore takes twice as long as the contraction. Because the animation lasts 6 seconds, the expansion takes 4 seconds, while the contraction takes 2 seconds.

Note that when you use `calcMode="paced"`, using more than two values has no effect if all values make the animation flow in the same direction. For example, consider the following attributes:

```
values="50;75;90;100;125;150" calcMode="paced"
```

Each value in the list above is greater than the preceding value. The animation therefore flows continuously in a positive direction. With the `paced` mode, though, only the first and last values will affect the speed. (This is not true with the `linear` value, however.) Hence, the preceding example functions the same as the following:

```
values="50;150" calcMode="paced"
```

Multiple values affect a paced animation only when they reverse the animation's direction. In the following example, the animation flows positively from the first to the second point, negatively from the second to the third point, and so on. Each value therefore affects the animation's appearance:

```
values="50;90;75;125;100;150" calcMode="paced"
```

Creating Additive and Cumulative Animations

As summarized in the following table, the `accumulate` and `additive` attributes let you create animations that build through repeating cycles.

Additive and Cumulative Attributes

Attribute	Value	Function	Reference
`accumulate`	`none|sum`	Makes a repeating animation build with each iteration.	page 356
`additive`	`replace|sum`	Adds the animation value to the existing value.	page 355

Adding Animation Values to a Base Value

Using additive animation, you can animate target attributes by increments, rather than by absolute values. You can use additive animations for sizes and

placements, but not colors. Although additive animations are useful on their own, they are more powerful when combined with cumulative animations, which the next section describes. To illustrate additive animation, consider the following region:

```
<region id="video_region" width="320" height="240"/>
```

Suppose that you want to add 60 pixels to the width of this region in three steps, each step adding 20 pixels to the width. As described in the preceding sections, you can do this by specifying the exact width values in a values list:

```
<animate targetElement="video_region" attributeName="width" dur="3s"
values="340;360;380" calcMode="discrete"/>
```

Or, you could use `additive="sum"` to tell RealONE Player to treat each specified value as an increment to add to the original width value:

```
<animate targetElement="video_region" attributeName="width" dur="3s"
values="20;40;60" additive="sum" calcMode="discrete"/>
```

The preceding example tells RealONE Player to add 20 pixels to the region's original width, then add 40 pixels to the original width, then add 60 pixels to the original width. Note that each list value is added to the target region's original width, not to the animated width created by the preceding list value.

Additive animation also works with the `from` and `to` values, but it provides little benefit. For example, you could add 20 pixels to a region width with these attributes:

```
from="0" to="20" additive="sum"
```

But it's simpler in this case to use just the `by` attribute:

```
by="20"
```

Making Animations Repeat and Grow

A cumulative animation uses a `repeatCount` or `repeatDur` attribute to repeat the animation. It also uses `accumulate="sum"` to increase or decrease the animated value with each iteration. You can use cumulative animations for sizes and placements, but not colors. To demonstrate cumulative animation, consider the following region:

```
<region id="image_region" width="180" height="180"/>
```

A simple cumulative animation could use the `by` attribute to increase the region width on each iteration of an animation. The following animation repeats four times. On each iteration, the region's width increases by 16 pixels:

```
<animate targetElement="image_region" attributeName="width" dur="2s"
by="16" accumulate="sum" repeatCount="4" calcMode="discrete"/>
```

The by attribute always adds a certain amount to the target value, making it naturally additive. When using a values list with a cumulative animation, though, you need to include additive="sum" as described in "Adding Animation Values to a Base Value" on page 355. Consider the following example:

```
<animate targetElement="image_region" attributeName="width" dur="2s"
values="5;20" additive="sum" accumulate="sum" repeatCount="2"
calcMode="discrete" fill="freeze"/>
```

In this example, each repetition lasts two seconds, the calcMode value is discrete, and there are two values (5 and 20) in the values list. These attributes cause the animation to behave as follows:

- When the animation starts, the region width instantly grows by 5 pixels. Because the region was defined to have a 180 pixel width, its new width is 185 pixels.
- One second after the animation starts, the region width grows to 200 pixels, 20 pixels more than its original width. Note that the 20 pixels are added to the original width, not to the preceding animated width. At this point, therefore, the region is 15 pixels wider than it was one second earlier.
- At two seconds, the animation repeats. Because the animation is cumulative, the region does not reset to its original width. Instead, it grows by an additional 5 pixels to a width of 205 pixels.
- At three seconds, the region grows 20 pixels wider than its width at the start of the second repetition. It therefore ends at a final width of 220 pixels.

Using the Specialized Animation Tags

The following sections describe the specialized animation tags, which are variations of the <animate/> tag. You should understand how the <animate/> tag works before using the specialized tags summarized in the following table.

Specialized Animation Tags

Tag	Function	Reference
<animateColor/>	Changes a color value for a region or clip.	page 358
<animateMotion/>	Moves an element both horizontally and vertically.	page 359
<set/>	Sets an attribute to a new value instantly.	page 360

For More Information: For information about the <animate/> tag and the various attributes you can use in the specialized animation tags, see "Creating Basic Animations" on page 345.

Animating Colors

The <animateColor/> tag works like an <animate/> tag, but is limited to color animations only. Although you can perform any color animation with an <animate/> tag, you may find it useful to use <animateColor/> to distinguish color animations from other animations. The following table lists the clip and region color properties you can animate. In other words, you can use the following attributes as values for attributeName in an <animateColor/> tag.

attributeName Values Used in the <animateColor/> Tag

Attribute	Target	Effect	Reference
backgroundColor	clip. region, window	Modifies a root-layout, secondary wndow, or region background color.	page 215
color	clip	Changes a <brush/> object color.	page 148

The following example changes a region's background color to red:

```
<animateColor targetElement="image_region" attributeName="backgroundColor"
to="red" begin="1s" dur="12s" fill="freeze"/>
```

You can also animate an element through several colors with a values list, as shown in the following example:

```
<animateColor targetElement="image_region" attributeName="backgroundColor"
values="red;blue;yellow" calcMode="discrete" .../>
```

When you use calcMode="discrete", only the specified colors display. If you use paced or linear as the calcMode value, though, intermediate colors display. The paced and linear modes create subtle differences when used with colors. Suppose you specify the following values:

```
values="#FFDD11;#FFDD22;#FFDDFF"
```

With the paced mode, the animation flows smoothly from #FFDD11 to #FFDDFF. With the linear mode, though, the first half of the animation flows from #FFDD11 to #FFDD22. The second half of the animation flows from #FFDD22 to #FFDDFF. Because the first half covers fewer color values, the color changes appear to occur more slowly than in the second half.

> **For More Information:** For more on calcMode, see "Controlling How an Animation Flows" on page 353. Appendix D describes SMIL color values.

> **Tip:** Color animations are not additive or cumulative, so do not use the by, additive, or accumulate attribute when animating a color with <animateColor/> or <animate/>.

Creating Horizontal and Vertical Motion

An <animateMotion/> tag moves an element both horizontally and vertically. Although you can move an element both horizontally and vertically by using two <animate/> tags, it's often simpler to use a single <animateMotion/> tag, which has the following features and restrictions that differentiate it from the <animate/> tag:

- The <animateMotion/> tag does not use an attributeName attribute. The tag always selects the upper-left corner of the element defined with the targetElement attribute.
- The <animateMotion/> tag can use a values list, or the to, by, and from attributes. You must always specify value pairs, the two values separated by a comma. The first value in the pair is the horizontal coordinate (x-coordinate), and the second value is the vertical coordinate (y-coordinate). The following are sample values:
  - to="120,180"
  - by="75%,15%"
  - values="60,120;80,150"

- The default value for calcMode is paced, rather than linear. This produces a smoother flow of motion when you animate an element through several points. For more on calcMode, see "Controlling How an Animation Flows" on page 353.

The following example moves the upper-left corner of the targeted region to the three points defined in the values list. Because calcMode="discrete" is used, the region will jump from point to point rather than flow smoothly:

```
<animateMotion targetElement="image_region" values="180,180;60,340;125,95"
calcMode="discrete" begin="7s" dur="5s" fill="freeze"/>
```

Setting an Attribute Value

The <set/> tag changes an attribute to a specific value until the tag is no longer active. It is useful, for example, to change a region background color while a clip plays. You can use this tag with any attribute that you can animate with the <animate/> tag. The <set/> tag can therefore change element sizes, positions, and colors. It uses fewer animation attributes than <animate/>, as summarized in the following table.

<set/> Tag Attributes

Attribute	Value	Function	Reference
attributeName	*attribute_name*	Selects the attribute to animate.	page 346
targetElement	*ID*	Identifies the tag that contains the animated attribute.	page 346
to	*pixels*|*percentage*|*color_value*	Sets the new attribute value.	page 351

A <set/> tag typically uses the basic SMIL timing attributes of begin, dur, and fill. The following example shows a <set/> tag changing a region background color. The region color changes instantly when the animation becomes active, then resets to its previous value after 30 seconds:

```
<set targetElement="video_region" attributeName="backgroundColor" to="blue"
dur="30s"/>
```

Manipulating Animation Timing

RealONE Player supports SMIL time manipulations for animations (and only animations). Time manipulations can control the rate of an animation to make it appear to accelerate or decelerate, for example. A later version of this

guide will explain how to create time manipulations. Currently, you can learn about time manipulations in the SMIL 2.0 specification:

http://www.w3.org/TR/smil20/smil-timemanip.html

CHAPTER

17

SWITCHING

SMIL switching gives you a powerful way to provide different clips that RealONE Player chooses between based on certain criteria. For example, you can have each RealONE Player select an audio track based on each viewer's language preference. This chapter explains how to set up switch groups to stream different clips to different audiences.

Understanding Switching

SMIL switching is a powerful means to tailor presentations for different audiences *without* making viewers choose which presentation they wish to view. In cases such as language choice, SMIL switching occurs automatically based on a preference the viewer has set in RealONE Player. In other cases, such as available bandwidth or monitor size, switching is based on an attribute that the viewer does not control. In all cases, however, RealONE Player automatically makes the choice without input from the viewer.

> **Tip:** When you want viewers to choose options themselves, use an <excl> tag, not a <switch> tag. For more on the <excl> tag, see "Creating an Exclusive Group" on page 186.

> **Note:** The following sections demonstrate switching with clips recorded in different languages. Keep in mind, though, that the same principles apply to switching through other criteria, such as bandwidth or monitor size.

Creating a Switch Group

A switch group starts with a <switch> tag and ends with a </switch> tag. Between these tags, you list multiple options, such as multiple clip source tags, that each contain a test attribute. RealONE Player evaluates the options in the order you list them, choosing the first option that it can play. For

example, in the following simple <switch> group, the systemLanguage test attributes cause RealONE Player to choose one of two audio clips based on its language preference:

```
<switch>
  <audio src="french.rm" systemLanguage="fr"/>
  <audio src="german.rm" systemLanguage="de"/>
</switch>
```

Only RealONE Players in which the viewer has selected French (fr) as the language preference will choose the first clip. Only RealONE Players in which the viewer has selected German (de) as the language preference will choose the second clip. A RealONE Player with another language preference will not play either clip: it simply ignores the clips in the <switch> group and proceeds to the next part of the presentation. Hence, a RealONE Player either plays just one clip from a <switch> group, or it plays no clip. But it never plays more than one option.

Adding a Default Option to a Switch Group

To reach the widest audience of viewers, a <switch> group needs to have a default option. Without this option, certain RealONE Players may not play any clips in the group. A default option must satisfy these criteria:

- The default option must not include a test attribute.

  Any RealONE Player will choose an option that does not include a test attribute. Hence, any RealONE Player that did not choose an option with a test attribute will choose the option without the test attribute.

- The default option must be the last option in the <switch> group.

  RealONE Player always evaluates options in the order they are listed, playing the first suitable option it finds. If you list a default option before options that includes test attributes, no RealONE Player will ever evaluate the test attributes of the options following the default option.

The following example modifies the preceding example to add an English-language clip as the default choice:

```
<switch>
  <audio src="french.rm" systemLanguage="fr"/>
  <audio src="german.rm" systemLanguage="de"/>
  <audio src="english.rm"/>
</switch>
```

Note that the English-language clip is listed last and does not include a systemLanguage test attribute, making it the default. In this example, any RealONE Player with a preference set to a language other than French or German will choose the English clip. For example, all RealONE Players with a language preference for Swedish, Korean, English, and so on choose the English-language clip.

Using Inline Switching

It is not always necessary to use a <switch> tag for switching. You can also use inline switching, which adds test attributes to clip source tags in a <par> or <seq> group. When RealONE Player encounters a test attribute, it evaluates the attribute to determine if it should play or skip the clip. In the following example of a parallel group, RealONE Player always plays the Flash clip, and then chooses the French or German audio clip based on its language preference:

```
<par>
  <ref src="cartoon.swf" region="playback"/>
  <audio src="french.rm" systemLanguage="fr"/>
  <audio src="german.rm" systemLanguage="de"/>
</par>
```

Choosing Inline Switching or a Switch Group

Although useful in many situations, inline switching cannot provide a default option, making it less powerful than a <switch> group. Consider the following example, which attempts to add a default English-language clip to the preceding example:

```
<par>
  <!-- This is NOT a good example of switching. -->
  <ref src="cartoon.swf" region="playback"/>
  <audio src="french.rm" systemLanguage="fr"/>
  <audio src="german.rm" systemLanguage="de"/>
  <audio src="english.rm"/>
</par>
```

The parallel group shown above will work for every RealONE Player, *except* those with a French or German language preference! For instance, a RealONE Player with a language preference for French plays the Flash (.swf) clip, the French RealAudio clip, *and* the English-language clip, which has no systemLanguage attribute that ties it to a language preference. Hence, the viewer hears an incomprehensible blend of French and English.

Only a <switch> tag causes RealONE Player to evaluate all options as a group and choose only one option. The following example illustrates the correct way to add the default, English-language choice to the parallel group shown above:

```
<par>
  <!-- This is a good example of switching. -->
  <ref src="cartoon.swf" region="playback"/>
  <switch>
    <audio src="french.rm" systemLanguage="fr"/>
    <audio src="german.rm" systemLanguage="de"/>
    <audio src="english.rm"/>
  </switch>
</par>
```

Available Test Attributes

The following table lists the test attributes available for switching. These attributes are described in detail in the remainder of this chapter.

Switch Attributes

Attribute	Value	Tests For	Reference
systemAudioDesc	on|off	audio descriptions preference	page 372
systemBitrate	*bits_per_second*	total available bandwidth	page 370
systemCaptions	on|off	captions preference	page 372
systemCPU	*CPU_type*	computer CPU type	page 373
systemLanguage	*language_code*	language preference	page 367
systemOperatingSystem	*OS_name*	computer operating system	page 374
systemOverdubOrSubtitle	overdub|subtitle	overdubbing or subtitle preference	page 369
systemScreenDepth	1|4|8|24|32	monitor color bit depth	page 376
systemScreenSize	*height*X*width*	monitor size	page 376

Tips for Writing Switch Groups

- The <switch> tag must fall within the <head> or <body> section of your SMIL file. In other words, the <smil>, <head>, or <body> tags cannot fall within a <switch> tag.

- You can use a <switch> tag in the header section to let RealONE Player choose between alternative layouts. See "Example 3: Resizing the Media Window for Captions" on page 381 for an example of modifying the layout to accommodate system captions.
- You can use a <switch> tag to switch between alternative <seq>, <par>, or <excl> groups. In these cases, you add the test attributes to the group tags instead of to the clip source tags.
- You can use more than one test attribute in a tag. For example, you might test for monitor size and color depth at the same time. When there are multiple test attributes, RealONE Player must satisfy all the attribute values before it chooses the clip. Alternatively, you can nest <switch> tags to achieve the same results. See "Switching with Multiple Attributes" on page 377 for examples of how to use both methods.
- In most cases, you'll want to create a default value so that every RealONE Player will find an option that it can play. In some cases, though, you may not want a default. If you're creating a group that switches between clips streaming at 300 Kbps, 200 Kbps, and 100 Kbps, for example, you may not want to include a default choice. That way, RealONE Players connected through dialup modems don't request any of the clips.
- In cases where you do not want certain RealONE Players to choose an option, you do not necessarily have to leave the default option out. Instead, you can use the default option to display a graphic or RealText clip informing viewers of the reason their RealONE Players cannot play the presentation.
- To switch between clips that use SMIL hyperlinks, create the links with <area/> tags inside the clip source tags, rather than with <a> and tags around the clip tags. Add the test attribute to the clip tag, as shown here:

```
<video src="video1.rm" systemLanguage="fr" ...>
  <area href="http://www.example.com" .../>
</video>
```

> **For More Information:** For more on the <area/> tag, see "Using the <area/> Tag" on page 283.

Switching Between Language Choices

When source clips are in different languages, use a test attribute of systemLanguage in the clip source tag or group tag. The following example

shows a video slideshow with separate audio narrations in French, German, Spanish, Japanese, Korean, and English. Each RealONE Player requests the same slideshow, but chooses an audio clip based on its language preference and its evaluation of the systemLanguage values:

```
<par>
  <ref src="seattle_slides.rp"/>
  <!-- select audio based on RealONE Player language preference setting -->
  <switch>
    <audio src="seattle_french.rm" systemLanguage="fr"/>
    <audio src="seattle_german.rm" systemLanguage="de"/>
    <audio src="seattle_spanish.rm" systemLanguage="es"/>
    <audio src="seattle_japanese.rm" systemLanguage="ja"/>
    <audio src="seattle_korean.rm" systemLanguage="ko"/>
    <audio src="seattle_english.rm"/>
  </switch>
</par>
```

The last audio option in the preceding example is the default. Because the last option does not have a test attribute, a RealONE Player that does not have French, German, Spanish, Portuguese, Japanese, or Korean set as its preferred language chooses the English clip.

Setting Language Codes

Appendix G lists the codes used as systemLanguage values. In some cases, a primary language code has variation codes. For instance, es is the primary code for Spanish, corresponding to Spanish as spoken in Spain. This code has several variations, such as es-mx for Mexican Spanish and es-pr for Puerto Rican Spanish. When variation codes are used, RealONE Player uses the following rules to select a clip:

- A RealONE Player with a preference for a language variation will choose either the variation code or the primary code, whichever comes first in the <switch/> group. For example, a RealONE Player with a preference for Mexican Spanish plays clips designated with es-mx or es. If a clip with the value es comes first, RealPayer does not continue to evaluate options to determine if the es-mx option is present.
- A RealONE Player with a preference for a primary language code will not choose clips that use variation codes. For example, a RealONE Player with a preference for Spanish as spoken in Spain chooses only clips designated

with es. If the only choices are es-mx and es-pr, for instance, this RealONE Player does not choose either option.

If you have different clips for different language variations, list the clip that corresponds to the primary code as the last option, as shown here:

```
<switch>
  <audio src="mexico.rm" systemLanguage="es-mx"/>
  <audio src="puertorico.rm" systemLanguage="es-pr"/>
  <audio src="defaultspanish.rm" systemLanguage="es"/>
</switch>
```

In the preceding example, RealONE Players with a preference for Mexican Spanish (es-mx) choose the first clip. RealONE Players with a preference for Puerto Rican Spanish (es-pr) choose the second clip. All other RealONE Players with a preference for any variation of Spanish choose the last clip. For instance, a RealONE Player with a preference for Chilean Spanish chooses the es option because its preferred variation (es-cl) is not listed.

Providing Subtitles or Overdubbing

For clips in foreign languages, RealONE Player viewers can set a preference for subtitles or overdubbing. The systemOverdubOrSubtitle attribute tests for this preference, displaying clips based on the viewer's choice. It can have one of two values, either overdub or subtitle. Suppose that you have three versions of a RealVideo clip:

1. an original French version (original.rm)
2. a version dubbed in English (dubbed.rm)
3. the original French version with English subtitles (titled.rm)

You can use systemOverdubOrSubtitle along with systemLanguage in a <switch> group as shown in the following example:

```
<switch>
  <!-- Version for RealONE Players with a preference for English and overdubbing. -->
  <video src="dubbed.rm" systemLanguage="en" systemOverdubOrSubtitle="overdub"/>
  <!-- Version for RealONE Players with a preference for English and subtitling. -->
  <video src="titled.rm" systemLanguage="en" systemOverdubOrSubtitle="subtitle"/>
  <!-- Version for RealONE Players with a language preference other than English. -->
  <video src="original.rm"/>
</switch>
```

In the preceding example, RealONE Players with a preference for English and overdubbing play the first clip. Any other RealONE Player preferring English plays the second clip. The original French clip is listed last with no systemLanguage attribute. This makes it the default played by RealONE Players that prefer French or another language besides English.

> **Note:** In the preceding example, the second clip does not need to specify subtitle explicitly. The systemOverdubOrSubtitle attribute uses only overdub or subtitle as its value. Because the first clip takes the overdub value, only the subtitle value is left for the second clip.

Switching Between Bandwidth Choices

To stream different clips to viewers at different connection speeds, use the systemBitrate test attribute to define options each RealONE Player can choose based on the total amount of bandwidth it has available. The systemBitrate attribute takes as a value the approximate bits per second required to stream the whole presentation. The following sample <switch> tag lists two different RealPix presentations. The first is for connections that have at least 80 Kbps of bandwidth. The second is for slower connections, down to 28.8 Kbps modems:

```
<switch>
  <ref src="slides1.rp" systemBitrate="80000"/>
  <ref src="slides2.rp" systemBitrate="20000"/>
</switch>
```

As shown above, list the bandwidth choices from fastest to slowest. RealONE Player evaluates options in order, selecting the first option it can play. If the 20,000 bps option were first, a RealONE Player with a high-speed connection would choose it because it is the first viable option. Also ensure that the last option satisfies the slowest connection speed you want to support. If the last choice is systemBitrate="60000", for example, RealONE Players on modems will not play the presentation because its bandwidth requirement is too high.

The more complex example below shows three sets of clips. Each <par> tag has a systemBitrate attribute that lists the approximate bandwidth the clips as a whole consume. Note that each group uses the same RealText clip, but has different RealAudio and RealPix clips created for its bandwidth:

```
<switch>
  <par systemBitrate="225000">
    <!--RealONE Players with 225 Kbps or faster connections choose this group-->
    <audio src="music1.rm"/>
    <ref src="slides1.rp" region="images"/>
    <textstream src="narration.rt" region="text"/>
  </par>
  <par systemBitrate="80000">
   <!--RealONE Players with connections between 80 and 225 Kbps get this group-->
    <audio src="music2.rm"/>
    <ref src="slides2.rp" region="images"/>
    <textstream src="narration.rt" region="text"/>
  </par>
  <par systemBitrate="20000">
    <!--RealONE Players with connections between 20 and 80 Kbps get this group-->
    <audio src="music3.rm"/>
    <ref src="slides3.rp" region="images"/>
    <textstream src="narration.rt" region="text"/>
  </par>
</switch>
```

For More Information: The table "Maximum Streaming Rates" on page 32 gives bandwidth guidelines for various network connections.

Switching with SureStream Clips

With RealAudio or RealVideo clips encoded for multiple bit rates with SureStream technology, you may or may not need to use the <switch> tag with a systemBitrate attribute. The following guidelines will help you to make this decision:

- When the presentation consists solely of a SureStream clip, simply link to that clip within the SMIL file. The clip then streams at the rate appropriate for RealONE Player's connection speed. You do not need to specify bandwidth choices with a <switch> tag.
- Use the <switch> tag when combining a SureStream clip with other clips encoded for single bandwidths. The SureStream clip is always used, but the <switch> group gives RealONE Player options for other clips. The following example illustrates a RealAudio SureStream clip and a choice between two RealPix presentations built for different bandwidths:

```
<par>
  <audio src="soundtrack.rm"/>
  <switch>
    <ref src="slideshow1.rp" systemBitrate="47000" region="images"/>
    <ref src="slideshow2.rp" systemBitrate="20000" region="images"/>
  </switch>
</par>
```

RealONE Players that have at least 47,000 bits per second of available bandwidth choose `slideshow1.rp`. If this slideshow takes 25 Kbps, for example, these RealONE Players pick a SureStream track from `soundtrack.rm` that requires 22 Kbps or less of bandwidth. RealONE Players with between 47,000 and 20,000 bps of available bandwidth choose `slideshow2.rp`, along with a SureStream track that keeps the combined clips under 20 Kbps.

> **For More Information:** For more on SureStream, see "SureStream RealAudio and RealVideo" on page 35. Refer to "Step 3: Develop a Bandwidth Strategy" on page 31 for information on targeting certain network connection speeds.

Enhancing Presentation Accessibility

RealONE Player users who are sight- or hearing-impaired can set an accessibility preference that gives them audio descriptions or captions, respectively, when those options are available. You can match RealONE Player viewers to these options with the `systemAudioDesc` and `systemCaptions` attributes. Both attributes, which you can use together or singly, take a value of either `on` or `off`. Suppose you have three versions of a video clip:

1. An original version for viewers with no accessibility preference (`video.rm`).
2. A version for sight-impaired viewers with a preference for audio descriptions (`video_descriptions.rm`).

   A video with audio descriptions might consist of a standard video that pauses intermittently while a separate audio track encoded in the clip describes upcoming scenes.
3. A version for hearing-impaired viewers with a preference for captions (`video_captions.rm`).

A video with captions might consist of a standard video that includes encoded captions similar to subtitles, but in the same language as the video's audio track.

You can use inline switching with the systemAudioDesc and systemCaptions attributes as shown in the following example to choose between clips based on the viewer's accessibility preference:

```
<seq>
  <video src="video_descriptions.rm" systemAudioDesc="on"/>
  <video src="video_captions.rm" systemCaptions="on"/>
  <video src="video.rm"/>
</seq>
```

For More Information: See "Displaying System Captions Using RealText" on page 379 for an example of using RealText to provide system captions.

Switching Based on the Viewer's Computer

Several <switch> tag attributes—systemCPU, systemOperatingSystem, systemScreenSize, and systemScreenDepth—let you switch between clips or groups based on the viewer's computer hardware or software. This lets you tailor a presentation's size or clip types, for example, based on the features of the machine running RealONE Player.

Switching for CPU Type

The systemCPU attribute lets you switch clips based on the processor for the RealONE Player computer. This attribute identifies the computer processor but no other machine attributes, such as the computer's clock speed, available memory, or operating system. The following table lists the possible values for the systemCPU attribute.

systemCPU Attribute Values

Attribute Value	Computer Processor Selected
alpha	Compaq Alpha processor
arm	Unix-based server processor
arm32	Unix-based server processor
hppa1.1	Hewlett-Packard Unix-based server processor

(Table Page 1 of 2)

systemCPU Attribute Values (continued)

Attribute Value	Computer Processor Selected
m68k	pre-PowerPC Macintosh
mips	Unix-based server processor
ppc	PowerPC Macintosh and Linux
rs6000	IBM Unix-based server processor
unknown	unknown processor type
vax	DEC VAX running VMS or Unix
x86	Intel chip set for Windows and Linux PCs and servers

(Table Page 2 of 2)

Note: The preceding table lists all systemCPU attribute values defined for SMIL. This does not mean, however, that RealONE Player is available for each hardware platform.

Switching for Operating System

The systemOperatingSystem attribute lets you switch clips based on the operating system running on the RealONE Player computer. This attribute does not discriminate between various versions of an operating system, however. The following table lists the values for systemOperatingSystem. The last column indicates if a version of RealONE Player is available for that operating system. Note, however, that RealONE Player availability is subject to change.

systemOperatingSystem Attribute Values

Attribute Value	Operating System Selected	RealONE Player?
aix	IBM AIX version of Unix	**yes**
beos	Be operating system	no
bsdi	Berkeley Software Design's version of Unix	no
dgux	Data General UX version of Unix	no
freebsd	FreeBSD version of Unix	no
hpux	HP-UX version of Unix	**yes**
irix	Silicon Graphics Irix version of Unix	**yes**
linux	Any Linux distribution	**yes**
macos	Any Macintosh operating system, including MacOSX	**yes**

(Table Page 1 of 2)

systemOperatingSystem Attribute Values (continued)

Attribute Value	Operating System Selected	RealONE Player?
ncr	NCR network operating system	no
nec	NEC version of Unix	no
netbsd	Network BSD version of Unix	no
nextstep	NeXT operating system	no
nto	NTO version of Unix	no
openbsd	Open BSD version of Unix	no
openvms	Open VMS	no
os2	IBM OS/2	no
osf	Open Software Foundation's version of Unix	no
palmos	Palm operating system	no
qnx	QNX Software System's realtime platform	no
rhapsody	Macintosh OSX Server	no
sco	Caldera version of Unix (fomerly Santa Cruz Operations)	no
sinix	Siemens Nixdorf version of Unix	no
solaris	Sun Solaris version of Unix	**yes**
sunos	Sun version of Unix pre-dating Solaris	no
unixware	Caldera version of Unix (formerly Novell)	**yes**
unknown	unknown operating system	n/a
win16	Microsoft Windows 16-bit OSes	**yes**
win32	Microsoft Windows 32-bit OSes	**yes**
win9x	Microsoft Windows 95/98/ME	**yes**
wince	Microsoft Windows CE and PocketPC	**yes**
winnt	Microsoft Windows NT/2000/XP	**yes**

(Table Page 2 of 2)

Switching for Monitor Size or Color Depth

Two test attributes, `systemScreenSize` and `systemScreenDepth`, let you switch clips based on the size and color capability of the monitor displaying RealONE Player. They are useful if you have different versions of the same video in different sizes or different color depths, for example.

Specifying a Monitor Size

The systemScreenSize attribute uses a pixel measurement value in the form *heightXwidth*. The value specifies that the monitor displaying RealONE Player must be of the given size or larger. The following are common systemScreenSize values:

1024X1280	common size for 21-inch monitors or larger
768X1024	common size for 17-inch monitors or larger
600X800	common size for 15-inch monitors or larger
480X640	smallest desktop monitor size in general use

> **Note:** You must use a capital "X". Note, too, that monitor sizes are commonly referred to in a width-by-height format, such as 640-by-480. With SMIL, though, you must specify height first.

Because a monitor must be at least the specified size for RealONE Player to choose an option, always list options from the largest to the smallest screen size as shown above. If you listed systemScreenSize="480X640" first, for example, all RealONE Players on standard desktop computers would choose that option because all standard desktop monitors are at least that size.

> **Tip:** Keep in mind that computer users can generally set their monitor resolutions differently. Some 17-inch monitors may have a resolution of 768X1024 for example, while others are set to 600X800.

Specifying a Color Depth

The systemScreenDepth attribute uses an integer value that specifies the color bit depth of the monitor. The monitor must have the given bit depth or higher to play the clip. The following are common systemScreenDepth values:

32	millions of colors
24	millions of colors
16	thousands of colors
8	256 colors
4	16 colors
1	black-and-white

Because a monitor must have at least the specified color depth for RealONE Player to choose an option, always list options from the highest bit depth to the lowest as shown above. If you listed `systemScreenDepth="8"` first, for example, all RealONE Players on standard color monitors would choose that option because all standard color monitors can display at least 256 colors.

Switch Group Examples

The following examples illustrate different ways to use switching. Note that there are many applications for switching, and many ways to write SMIL presentations that include switching. To view more examples, get the HTML+Javascript version of this guide as described in "How to Download This Guide to Your Computer" on page 5, and view the **Sample Files** page.

Switching with Multiple Attributes

Using multiple test attributes in a `<switch>` group, you can have RealONE Player choose clips based on combined criteria, such as both available bandwidth and language preference. There are two ways to do this:

- include multiple test attributes in each tag
- nest `<switch>` groups

Example 1: Multiple Test Attributes

In the following example, the first two RealAudio clips have two test attributes each—one for language and one for bandwidth. Both attributes must be viable for RealONE Player to choose the clip. Because RealONE Player evaluates the `<switch>` choices from top to bottom, selecting the first viable option, the last two choices do not have language attributes. This lets all RealONE Players other than those with French selected as their language preference choose between the two English-language clips, based on their available bandwidth:

```
<switch>
  <!-- French language choices -->
  <audio src="french2.rm" systemLanguage="fr" systemBitrate="47000"/>
  <audio src="french1.rm" systemLanguage="fr" systemBitrate="20000"/>
  <!-- English language choices (default) -->
  <audio src="english2.rm" systemBitrate="47000"/>
  <audio src="english1.rm" systemBitrate="20000"/>
</switch>
```

Example 2: Nested <switch> Groups

The next example adds RealText clips in both French and English to the presentation possibilities. Here, <switch> groups are nested so that RealONE Players with French set as their language preference play the French RealText clip and choose from the set of French-language RealAudio clips, based on available bandwidth. All other RealONE Players play the English RealText clip and choose from the set of English-language RealAudio clips:

```
<switch>
  <!-- Choose French as the language. -->
  <par systemLanguage="fr">
    <textstream src="frenchcredit.rt" region="credits_region" fill="remove"/>
    <switch>
      <!-- Choose fast or slow bit rate for French audio -->
      <audio src="french2.rm" systemBitrate="47000"/>
      <audio src="french1.rm" systemBitrate="20000"/>
    </switch>
  </par>
  <!-- Choose English as  the language. This is the default. -->
  <par>
    <textstream src="enlgishcredits.rt" region="credits_region" fill="remove"/>
    <switch>
      <!-- Choose fast or slow bit rate for English audio -->
      <audio src="english2.rm" systemBitrate="47000"/>
      <audio src="english1.rm" systemBitrate="20000"/>
    </switch>
  </par>
</switch>
```

Switching for Different Video Sizes

As described in "Different Clip Dimensions for Different Bandwidths" on page 62, you can encode different sizes of the same video, streaming a small clip over slow modems and a larger clip (or clips) over faster connections. Reducing the video size for slower connections ensures that the video's frame

rate and visual quality remain high. For example, you could create the three clips listed in the following table.

RealVideo Clips at Different Sizes

Clip Name	Dimensions	SureStream Audiences	systemBitrate value
videosmall.rm	176 x 132	28.8 and 56 Kbps Modems	20000
videomedium.rm	240 x 180	ISDN and corporate LANs	45000
videobig.rm	320 x 240	256, 384, and 512 Kbps DSL and cable modems	225000

In the following example, each <switch> tag test attribute uses the target bit rate of its clip's slowest SureStream stream. The <switch> tag then presents the three RealVideo choices to RealONE Player from fastest to slowest:

```
<switch>
  <video src="videobig.rm" systemBitrate="225000" region="video_region" .../>
  <video src="videomedium.rm" systemBitrate="45000" region="video_region" .../>
  <video src="videosmall.rm" systemBitrate="20000" region="video_region" .../>
</switch>
```

For More Information: Target bit rates are listed in the table "Maximum Streaming Rates" on page 32.

Displaying System Captions Using RealText

As the section "Enhancing Presentation Accessibility" on page 372 explains, you can use the systemCaptions attribute to display captions for hearing-impaired viewers. The following examples demonstrate various ways to display RealText captions coordinated with an audio track.

Note: You can use any means to provide captions. The systemCaptions="on" attribute simply tells RealONE Player to play a certain clip if the viewer has set the captions preference. There are no requirements for what type of clip to use for captions, though.

For More Information: Chapter 6 explains how to write and time a RealText clip. See Chapter 11 for information about layouts.

Example 1: Using a Transparent RealText Overlay

Although it may not be suitable in all cases, the simplest way to provide captioning is to overlay a clip with a RealText clip that has a transparent background. To do this, you define two regions, one for the video, and one for the captions, using the z-index attribute to ensure that the captions appear in front, as shown in the following example:

```
<layout>
  <root-layout width="320" height="240" backgroundColor="black"/>
  <region id="video_region1" z-index="1"/>
  <region id="text_region" height="40" bottom="0" left="10" z-index="2"/>
</layout>
```

You then play the RealText clip in parallel with the main clip, using systemCaptions="on" to display the RealText clip only in RealONE Players that have a preference for system captions. Because system captions are either on or off, you can easily use inline switching (no <switch> tag), as shown here:

```
<par>
  <video src="video.rm" region="video_region1" fill="remove"/>
  <textstream src="transparentcaptions.rt" region="text_region"
    systemCaptions="on" fill="remove"/>
</par>
```

Example 2: Creating a Captions Region

If you do not want to overlay the video as described in the preceding example, you can create a separate region for the captions through your SMIL file layout. The following layout is similar to that used in the preceding example, except that the captions region appears below the video region rather than on top of it:

```
<layout>
  <root-layout width="320" height="300" backgroundColor="black"/>
  <region id="video_region" height="240"/>
  <region id="text_region" height="40" top="260" left="10"/>
</layout>
```

If you play a video in parallel with a captions clip as shown in the preceding example, the captions region would appear blank for viewers who have the captions preference turned off. Alternatively, you can create a "filler clip" that displays in the captions region when captions are off. This clip might simply thank the viewer for watching the presentation. The following example demonstrates how to do this:

```
<par>
  <video src="../clips/video3.rm" region="video_region" fill="remove"/>
  <switch>
    <textstream src="videocaptions.rt" region="text_region" systemCaptions="on"/>
    <textstream src="fillercaptions.rt" region="text_region" systemCaptions="off"/>
  </switch>
</par>
```

Example 3: Resizing the Media Window for Captions

This example demonstrates how to use systemCaptions in <layout> tags to change layouts depending on whether or not captions are displayed. The following layout creates a captions region only when captions are turned on. Note that in each layout, the video region has a unique ID, which is required by SMIL. But both video regions have the same name:

```
<switch>
  <layout systemCaptions="on">
    <root-layout width="320" height="300" backgroundColor="black"/>
    <region id="video_region1" regionName="video" height="240"/>
    <region id="text_region" height="40" top="260" left="10"/>
  </layout>
  <layout systemCaptions="off">
    <root-layout width="320" height="240" backgroundColor="black"/>
    <region id="video_region2" regionName="video"/>
  </layout>
</switch>
```

> **Tip:** Although the preceding example uses systemCaptions in the <layout> tag, you could use the attribute in <root-layout/> and <region/> tags instead to display or hide individual regions based on RealONE Player's captions setting.

In the SMIL body, you then assign clips to the regions. Note that the following markup assigns the single video clip to a region through the region name instead of the region ID. If you didn't use the region name, you'd need to create two <video/> tags, one assigned to video_region1, the other assigned to video_region2. Each tag would require a systemCaptions attribute to turn the tag on or off depending on the captions preference. With the following method, only the RealText clip uses the systemCaptions attribute:

```
<par>
  <video src="video.rm" region="video" .../>
  <textstream src="captions.rt" region="text_region" systemCaptions="on" .../>
</par>
```

For More Information: See "Setting Region IDs and Names" on page 207 for more on region names.

CHAPTER

18

PREFETCHING

Prefetching allows you to manage bandwidth in a complex presentation. This helps you to ensure that the presentation streams smoothly. You can stream data for high-bandwidth clips while low-bandwidth clips play, for example. To use prefetching, though, you must thoroughly understand how clips use bandwidth, as well as how to create a presentation timeline.

> **For More Information:** To learn more about bandwidth use, read Chapter 2. Chapter 12 describes the basics of SMIL timing.

Understanding Prefetching

Prefetching is a powerful feature for managing bandwidth in a streaming presentation. It lets you stream portions of large clips, or all data for small clips, before the clips play. RealONE Player stores the prefetched data in memory until clip playback begins. Using prefetched data, RealONE Player can display clips faster when they begin to play. This can reduce or eliminate the buffering that normally occurs when clips start to play.

Uses of prefetching include downloading small image files. If a presentation contains graphic buttons that display while a video plays, for example, you can prefetch the graphics files before the video begins. When the video-with-buttons segment starts, the graphics do not compete with the video for bandwidth. Another use of prefetching is to download an audio or video clip's preroll, which is described in the section "Buffering" on page 31, before the clip plays.

Prefetching data is useful only when streaming across a network. It has no discernible effect when clips reside on the viewer's local computer. To use it effectively, you typically need to have a presentation in which low-bandwidth sections precede high-bandwidth sections. In these cases, prefetching lets you take advantage of low bandwidth use to download data for upcoming high-

bandwidth segments. When you stream only a video, for example, prefetching offers no advantages. If a RealText clip precedes the video, though, you can use prefetching to stream the RealVideo clip's preroll while the RealText clip plays.

> **Warning!** You should have a strong understanding of timelines and bandwidth management when prefetching clip data. Incorrect use of this feature may stall your presentation or cause RealONE Player to use excessive amounts of memory.

Using the <prefetch/> Tag

To prefetch data, you use a <prefetch/> tag, which is similar to a clip source tag like <video/>. Instead of playing a clip, though, the <prefetch/> tag downloads all or part of the clip data for playback later. As with a clip source tag, the <prefetch/> tag uses a URL to indicate the data to download, and can include timing attributes such as dur. Unlike clip source tags, a <prefetch/> tag has its own attributes that govern the speed and amount of data downloaded. In the following example, a <prefetch/> tag downloads a video clip's preroll to RealONE Player while a RealText clip plays presentation credits:

```
<seq>
  <!-- Segment 1: Roll RealText credits and download video preroll. -->
  <par endsync="id(credits)">
    <texstream src="rtsp://realserver.example.com/credits.rt" id="credits" .../>
    <prefetch src="rtsp://realserver.example.com/video1.rm" mediaTime="15s"
      bandwidth="18000" />
  </par>
  <!-- Segment 2: Play the video. -->
  <video src="rtsp://realserver.example.com/video1.rm" region="main"/>
</seq>
```

In this example, the <prefetch/> tag downloads the first 15 seconds of the clip video1.rm at a rate of approximately 18 Kbps. RealONE Player holds this data in memory until the video plays, eliminating the buffering that occurs when the clip starts to play. RealONE Player matches the prefetched data to the video through the identical URLs in the <prefetch/> and <video/> tags.

Note that in the preceding example, the <par> tag has an endsync attribute that ends the group when the RealText clip finishes. Without this attribute, there could be empty playback time if the prefetching does not complete before the RealText clip finishes. Because RealONE Player treats a <prefetch/> clip like other source clips when it determines presentation timing, always use a timing

mechanism, such as endsync or dur, to ensure that prefetching does not interfere with presentation playback.

The following table summarizes the <prefetch/> tag attributes that control how much clip data is downloaded. The following sections describe how to use these attributes effectively.

<prefetch/> Attributes

Attribute	Value	Default	Function	Reference
bandwidth	*bps*|*percentage*	100%	Sets the bandwidth used to get data.	page 385
mediaSize	*bytes*|*percentage*	100%	Specifies the amount of data to prefetch based on the clip's size. Overrides mediaTime.	page 387
mediaTime	h|min|s|ms|*percentage*	100%	Sets the amount of data to prefetch based on the clip's duration.	page 388

Managing Prefetch Bandwidth

The <prefetch/> tag's bandwidth attribute governs how much bandwidth is assigned to fetching the clip's data. If you do not include the bandwidth attribute, prefetching uses all of the connection's available bandwidth, which is rarely desirable. You can specify a specific streaming speed in bits per second (bps), or indicate a percentage of the available bandwidth. Note that you can prefetch data at any bandwidth, regardless of the clip's normal streaming speed. For an audio clip that normally streams at 20 Kbps, for instance, you could prefetch data at any speed, from 1 Kbps to 100 Kbps or faster.

Specifying Prefetch Bandwidth in Bits Per Second

To specify the exact streaming speed in bits per second, start with the maximum recommended bandwidth for your slowest targeted connection. If 56 Kbps modems are your lowest-speed targets, for instance, use a 34 Kbps maximum streaming speed, as given in the table "Maximum Streaming Rates" on page 32. Next, determine how much bandwidth you can dedicate to prefetching. If you want to stream the prefetched data in parallel with a 16 Kbps RealAudio clip, for example, you have a maximum of 18 Kbps for prefetching:

```
<par endsync="id(music)">
  <audio src="..." id="music" dur="50s"/>
  <prefetch src="..." begin="10s" bandwidth="18000" mediaSize="20480"/>
</par>
```

In this example, data is prefetched at approximately 18 Kbps until either 20 Kilobytes of data have been received, or the audio clip stops playing. Note that the <prefetch/> tag's begin time means that the prefetching begins 10 seconds after the audio clip starts to play. This dedicates all available bandwidth to the audio clip during the first 10 seconds of playback, making the audio clip's own preroll stream faster. Although a begin value is optional, including it can help to manage bandwidth in the segment that includes prefetching.

Specifying Prefetch Bandwidth as a Percentage

Determining a percentage value to use for the bandwidth attribute is more complicated than specifying a specific bandwidth. It has useful benefits, though. Suppose that you use a bandwidth="50%" value when prefetching clip data. Over a 56 Kbps modem, the prefetching uses about 17 Kbps. Over a 256 Kbps DSL line, though, the prefetching uses over 100 Kbps, finishing much faster. If you used bandwidth="17000" instead, the prefetching would take place at the same rate over both connections.

The value you specify equates to a percentage of the usable bandwidth that RealONE Player detects, which may differ from the speeds listed in the table "Maximum Streaming Rates" on page 32. For a 56 Kbps modem, for example, the detected bandwidth will likely be higher or lower than the maximum streaming speed of 34 Kbps. But it will definitely be less than the modem's raw speed of 56 Kbps. Because you don't know the exact prefetching speed when you use a percentage value, you need to decide upon a value carefully.

To select a percentage value, start with the maximum streaming speed for your slowest target connection. If your slowest targets are 56 Kbps modems, use a 34 Kbps maximum streaming speed. Then determine how much bandwidth is left for prefetching. For instance, you have 18 Kbps available for prefetching data while a 16 Kbps RealAudio clip plays. This 18 Kbps is approximately 53 percent of the 34 Kbps maximum speed. However, because the speed RealONE Player detects may be higher or lower, select a lower percentage value, such as 45 percent, as shown in the following example:

```
<par endsync="id(music)">
  <audio src="..." id="music" dur="50s"/>
  <prefetch src="..." begin="10s" bandwidth="45%" mediaSize="20480"/>
</par>
```

Controlling Prefetch Data Download Size

Two attributes for the <prefetch/> tag, mediaSize and mediaTime, control the amount of data that RealONE Player downloads for each clip. Use just one of these attributes in each <prefetch/> tag. If you use both mediaSize and mediaTime, the mediaTime attribute is ignored. If you do not use either of these attributes, RealONE Player attempts to prefetch all the clip's data, which can cause RealONE Player to run out of memory with large clips such as videos.

> **Tip:** The amount of data you can prefetch depends on the amount of computer memory available to RealONE Player. To reach the widest audience, do not try to prefetch more than one Megabyte of clip data.

Prefetching a Specific Amount of Data

The mediaSize attribute allows you to set how much of the clip data to prefetch based on the clip's file size. You must use mediaSize rather than mediaTime for clips that do not have internal timelines, such as images. Specify the mediaSize value in bytes, or as a percentage of the clip's total size.

Specifying mediaSize in bytes and bandwidth in bits per second lets you determine exactly how long the prefetching lasts. The following example prefetches 10 Kilobytes of clip data at a rate of approximately 6 Kilobits per second. The prefetching therefore takes approximately 13.7 seconds to complete:

```
<prefetch src="..." mediaSize="10240" bandwidth="6000"/>
```

> **Note:** Remember, the bandwidth attribute is in *bits* per second, whereas the mediaSize attribute is in *bytes* (8 bits = 1 byte).

If you want to prefetch entire clips, such as whole GIF files, specify mediaSize="100%" or leave the attribute out of the <prefetch/> tag. In these cases, you'll need to know the size of the prefetched clip to determine how long the prefetching lasts.

Prefetching a Specific Length of a Clip's Timeline

For clips that have internal timelines, such as RealAudio, RealVideo, or Flash, you can use `mediaTime` instead of `mediaSize` to prefetch a specific stretch of the clip's timeline. This is useful for prefetching the clip's preroll. You can specify a percentage value, or a timing value as described in "Specifying Time Values" on page 239. The following example prefetches 10 seconds of clip data:

```
<prefetch src="..." mediaTime="10s" bandwidth="4000"/>
```

Keep in mind that `mediaTime` does not control how long prefetching lasts. The amount of time required for prefetching depends on the amount of data downloaded and the bandwidth. If the clip in the preceding example normally streams at 16 Kbps, for example, RealONE Player needs approximately 40 seconds to prefetch the first 10 seconds of the clip. This is because the prefetching bandwidth is only a quarter of the clip's streaming bandwidth.

> **Tip:** To determine how much preroll a clip requires, open the clip in RealONE Player, and use the **View>Clip Source** to view clip information.

Tips for Prefetching Data

The following sections provide additional pointers for using `<prefetch/>` tags to stream clip data.

RealAudio and RealVideo Prefetching

- You cannot prefetch a single stream of a SureStream RealAudio or RealVideo clip. If you specify `mediaTime="10s"`, for example, you will get the first ten seconds of every stream in the SureStream clip.
- For large streaming clips such as RealAudio, RealVideo, do not prefetch much more than the clip's preroll, which is typically 5 to 15 seconds. You can do this easily with an attribute and value such as `mediaTime="10s"`. Prefetching more data wastes bandwidth and can cause RealONE Player to run out of memory.
- Try not to prefetch data too far in advance of when the clip plays. There's little advantage to prefetching a video's preroll 15 minutes before it plays because viewers may stop the presentation within those 15 minutes. As well, RealONE Player has to reserve memory for that clip data for 15

minutes, and that memory may be more effectively used for rendering the clips that do play during that time span.

- Using constant bit rate encoding (CBR) or variable bit rate encoding (VBR) with a RealVideo clip does not affect prefetching, other than that VBR clips typically have a longer preroll than CBR clips. For more on VBR, see "Variable Bit Rate Encoding" on page 73, as well as your RealProducer documentation.
- Because of its low bandwidth requirements, RealText makes an ideal clip to display as you prefetch data for high-bandwidth clips such as RealAudio and RealVideo. See Chapter 6 for more information about RealText. Flash animation, which is described in Chapter 5, can also stream effectively at low bandwidths to mask prefetching.

Prefetch URLs

- You can use a `<meta name="base" content="URL"/>` tag with prefetching to set the base URL for all clips. See "Creating a Base URL" on page 152 for more information.
- Because RealONE Player matches prefetched data to clips based on URLs, you generally should not use prefetching when URLs are dynamically generated and may change. An example of this is a banner ad in which the URL changes each time the ad is requested from an ad server.
- Prefetching is compatible with the CHTTP caching protocol. For an example that demonstrates these two features, see "Prefetching and Caching an Image" on page 391.

SMIL Timing with Prefetching

- RealONE Player discards prefetched data after the clip plays the first time, even if the clip uses a `repeatDur` or `repeatCount` attribute. If the clip plays again later in the presentation, you need to prefetch its data again. Small files, though, can be cached. For an example of this, see "Prefetching and Caching an Image" on page 391.
- If you plan to use a `clipBegin` attribute to play a clip from some point other than its normal starting point, use the same `clipBegin` value in the `<prefetch/>` tag. For more on this attribute, see "Setting Internal Clip Begin and End Times" on page 242.

- RealONE Player can prefetch clips from any server, including Web servers. However, clipEnd and clipBegin attributes do not function for clips on Web servers. For more information, see "Limitations on Web Server Playback" on page 428.
- A <prefetch/> tag can have an ID like any clip source tag. This lets you use endsync to end a group when prefetching finishes, as explained in the example "Displaying an Image Until Prefetching Completes" on page 390. For basic information about IDs, see "SMIL Tag ID Values" on page 138.

Prefetch Testing

- Prefetching data is useful only when streaming across a network. It has no discernible effect when all clips reside on the viewer's local computer or on a CD, for example.
- Although playing the SMIL presentation on your local computer will help you catch SMIL syntax errors, it does not guarantee that prefetching is achieving the results you desire.
- When you use <prefetch/>, test your presentation by streaming it over a network at your target connection's bandwidth (by dialing in on a 56 Kbps modem, for example).

Prefetching Examples

The following examples show different ways to use prefetching.

Displaying an Image Until Prefetching Completes

In the following example, standby.gif is a small image file that asks the viewer to wait while the presentation loads. The endsync attribute that targets the <prefetch/> tag makes the image display until the video prefetching has completed. The <prefetch/> tag's begin time gives 100% of the available bandwidth to the GIF download for five seconds. After that, the prefetching takes almost all of the usable bandwidth:

```
<seq>
  <!-- Segment 1: Standby. -->
  <par endsync="id(fetchvid)">
    <img src="http://www.example.com/standby.gif" fill="freeze" region="main"/>
    <prefetch src="rtsp://realserver.example.com/video.rm" id="fetchvid"
      bandwidth="95%" mediaTime= "15s" begin="5s"/>
```

```
 </par>
 <!-- Segment 2: Play video. -->
  <video src="rtsp://realserver.example.com/video.rm" region="main"/>
</seq>
```

Prefetching and Caching an Image

The section "Caching Clips on RealONE Player" on page 154 explains how to store clips in the RealONE Player cache for later use. While caching and prefetching are different activities, they can be used together effectively to download and retain small files that are used repeatedly. Caching should never be used with large clips, however, because RealONE Player's cache is only a few Megabytes in size.

The following SMIL sample, which omits layout attributes, prefetches a GIF image used as the background for two videos. Because the CHTTP protocol is used, the image is cached and does not need to be prefetched a second time:

```
<seq>
 <!-- Segment 1: Play introductory section and download background. -->
 <par endsync="id(credits)">
  <textstream src="rtsp://realserver.example.com/credits.rt" id="credits" .../>
  <prefetch src="chttp://realserver.example.com/image1.gif"
   bandwidth="10000"/>
 </par>
 <!-- Segment 2: Play video 1 against background. -->
 <par>
  <video src="rtsp://realserver.example.com/video1.rm" .../>
  <img src="chttp://realserver.example.com/image1.gif" fill="freeze"/>
 <par>
 ... other segments ...
 <!-- Segment 6: Play video 2 against background. -->
 <par>
  <video src="rtsp://realserver.example.com/video2.rm" .../>
  <img src="chttp://realserver.example.com/image1.gif" fill="freeze"/>
 <par>
</seq>
```

> **Note:** Keep in mind that prefetching stores clip data in memory until the clip plays. Caching stores a copy of the clip on the computer's hard disk. That copy may remain in the cache for several hours or even days.

PART

VII

STREAMING YOUR CLIPS

Your hard work doesn't pay off until you've streamed your clips to others.Chapter 19 explains the option of embedding your presentation in a Web page. Chapter 20 provides step-by-step instructions for moving your streaming presentation to a server and linking your Web page to it.

CHAPTER
19

WEB PAGE EMBEDDING

With embedded playback, you can weave your clips through your Web page's text and graphics, and add controls such as stop and start buttons. It's as if you took RealONE Player apart and placed its pieces at different spots on your page. This chapter explains how to add markup to a Web page so that people can view your streaming presentation directly through their Web browsers.

Understanding Web Page Embedding

To embed streaming media in your Web page, you first produce your clip or clips. You can even use SMIL to put together an entire presentation containing multiple clips. You then embed your presentation by adding `<EMBED>` tags to your Web page, and using HTML commands to place your clips and various RealONE Player controls on your page.

<EMBED> Tags

You add to your Web page one `<EMBED>` tag for each playback window you want in your page, and one `<EMBED>` tag for each control, such as a **Stop** button, that you want to include. When you use `<EMBED>`, your presentation will work in browsers that support the Netscape plug-in architecture, including the following:

- Netscape Navigator 3.0 and later.
- Microsoft Internet Explorer 3.0 and later.

> **Note:** Even when you use the `<EMBED>` tag, RealONE Player communicates with the Internet Explorer browser using ActiveX technology. This makes the Netscape `<EMBED>` tag compatible with both major browsers.

<EMBED> Tag Syntax

<EMBED> tags are an extension of HTML. Because they are not SMIL tags, they do not use the same syntax rules as SMIL. The <EMBED> tag parameters are typically in this form:

```
PARAMETER=value
```

Parameter names can be any case, although this manual shows them uppercase. Except for file names, which typically must be lowercase, parameter values are not case-sensitive. Unless they are URLs, parameter values do not need to be enclosed in quotation marks.

Layout Possibilities

When you embed a presentation, you use HTML to structure your Web page and define where each streaming clip and RealONE Player control appears. A common practice is to define an HTML table, embedding clips and RealONE Player controls in various table cells. When you embed a SMIL presentation, you can define a layout using SMIL and HTML, or just HTML alone.

Defining a Layout with SMIL and HTML

As described in Chapter 11, you can use SMIL to define an overall media window size (the root-layout). You might create a layout that is 400 pixels wide by 300 pixels high, for example, and define smaller regions within that main area for clips. You then embed the entire playback area within your Web page using a single <EMBED> tag, adding RealONE Player controls around it with separate <EMBED> tags. All clips then appear within that 400-by-300 pixels area, just as they would when played in RealONE Player. In fact, your SMIL file can play in both your Web page and RealONE Player.

For More Information: See "Defining the Layout with SMIL" on page 412.

Defining a Layout with HTML Alone

You can leave layout information out of your SMIL file, and use SMIL simply to define your presentation timing and other playback features. In your Web page, you then create a separate <EMBED> tag for each clip, placing each clip anywhere on your page. In this case, all clips do not need to appear within a rectangular root-layout area. This gives you more layout flexibility than when defining the overall clip layout through SMIL. However, because your SMIL

file lacks layout information, it may have unexpected layout results if played directly in presentation.

> **For More Information:** See "Defining the Layout with HTML" on page 413.

RealONE Player Controls

In addition to clips, you can embed many different RealONE Player buttons, sliders, and information panels in your Web page. You might include separate start, stop, and pause buttons in your Web page, for example. Or, you could add entire control panels that contain multiple buttons and readouts. You can make these controls any size you want, too, giving you even more layout flexibility. The section "Adding RealONE Player Controls" on page 401 explains all the available controls.

Javascript and ActiveX

RealONE Player supports Javascript, which enables you to extend the `<EMBED>` tag capabilities. You can use Javascript to turn your own graphic image into a RealONE Player **Play** button, for example. RealONE Player also includes an ActiveX control that provides playback capabilities for the following products:

- Microsoft Internet Explorer 3.0 and later.
- Any application that supports ActiveX controls, such as Microsoft Visual Basic, Microsoft Visual C++, Microsoft Access, and so on.

This production guide does not explain these capabilities. For information on using Javascript or ActiveX with an embedded presentation, see the *RealPlayer Javascript and ActiveX Reference Guide*, which is available for download from the following Web page:

http://service.real.com/help/library/encoders.html

Using <EMBED> Tags

Each <EMBED> tag has three required parameters, and can include many optional parameters, which are described throughout this chapter. The following table lists the parameters to include in every <EMBED> tag.

Basic <EMBED> Tag Parameters

Parameter	Value	Function	Reference
HEIGHT	*pixels*	Sets the height of the clip or control.	page 400
NOJAVA	true|false	Keeps the Java virtual machine turned off.	page 400
SRC	*filename*.rpm	Gives URL to the Ram file (.rpm).	page 398
WIDTH	*pixels*	Sets the width of the clip or control.	page 400

A basic <EMBED> tag looks like the following, which creates a playback area 320 pixels wide by 240 pixels high within the Web page:

```
<EMBED SRC="presentation.rpm" WIDTH=320 HEIGHT=240 NOJAVA=true>
```

Specifying the Source

You must include the SRC parameter in every <EMBED> tag, even when the tag embeds a RealONE Player control instead of a clip. However, you don't specify a clip or SMIL file directly with SRC. Instead, you specify a Ram file that has a .rpm extension. This causes the browser to use RealONE Player as a helper, rather than launch it as a separate application. The .rpm file is a simple text file that gives the full URL to your clip or SMIL file.

> **For More Information:** For background on Ram files, see "The Ram File" on page 415.

Developing Your Presentation

The easiest means for developing your embedded presentation is to keep your clips in the same folder as your Web page on your desktop computer. Your <EMBED> tag can then link to a .rpm file in that folder:

```
<EMBED SRC="presentation.rpm" WIDTH=300 HEIGHT=134>
```

To embed a single video, for example, the .rpm file simply contains a local file URL to the clip (the file:// protocol designation is required):

```
file://video.rm
```

Delivering Your Presentation

When you are ready to deliver your presentation to your audience, move your files to their respective servers and change the URLs in your files.

Keeping the .rpm File and the Web Page Together

If you plan to keep the .rpm file with the Web page, you do not need to change the SRC values in your <EMBED> tags. You can simply transfer your .rpm file and your Web page to the same directory on your Web server.

Putting the .rpm File and the Web Page in Different Locations

If you move the .rpm file to a different directory than that Web page, link each <EMBED> tag's SRC parameter to the .rpm file with a full HTTP URL:

```
SRC="http://www.example.com/media/sample.rpm"
```

Linking to Streaming Clips

No matter where you put your .rpm file and your clips, modify the .rpm file to give the fully-qualified URL to the embedded clip or SMIL file. If the clip or SMIL file is on a Web server, use an HTTP URL. If the clip or SMIL file is on RealServer, use an RTSP URL.

> **Tip:** Always use a full URL in the .rpm file, even if all files and clips are in the same directory on a Web server. RealONE Player uses the .rpm file to locate the clip or presentation. Without a fully-qualified URL, RealONE Player must construct the location from the original Web page URL and the information in the .rpm file. This creates more possibility for errors.

> **For More Information:** For more information on URLs in .rpm files, see "Launching RealONE Player with a Ram File" on page 420. The section "The Difference Between RTSP and HTTP" on page 417 explains why RealServer uses the RTSP protocol instead of a Web server's HTTP protocol.

Linking to Local Clips

If you will make your presentation available to people on their local machines (through a download or a CD, for instance), you do not need to change any URLs from those described in "Developing Your Presentation" on page 398. In rare cases, though, you may want to use an absolute link, rather than a relative link, in the .rpm file. When using absolute links, use forward slashes in paths to create "Web style" paths. For example, instead of this URL:

file://c:\media\first.rpm

use this URL:

file://c:/media/first.rpm

Using RealServer's Ramgen to Eliminate the Ram File

When your embedded clips reside on a RealServer that uses the Ramgen feature, you can eliminate the .rpm file when you deliver your presentation. Your SRC parameter uses an HTTP URL to the clip or SMIL file on RealServer, and includes a /ramgen/ parameter along with the ?embed option:

```
SRC="http://realserver.example.com:8080/ramgen/sample.smil?embed"
```

The HTTP protocol is required because a browser cannot make an RTSP request. When /ramgen/ is used, though, RealServer causes the browser to start RealONE Player as a helper application, then streams the clip or SMIL file to RealONE Player using RTSP. Consult your RealServer administrator for the correct URL to your RealServer.

For More Information: For details on using the Ramgen option, see "Using Ramgen for Clips on RealServer" on page 425.

Setting the Width and Height

Required for each <EMBED> tag, the WIDTH and HEIGHT parameters set the size of the playback area. If you omit these parameters, the playback area may appear as a tiny icon because streaming media presentations do not resize themselves automatically. The values for WIDTH and HEIGHT are in pixels by default, so a width of 300 creates a playback area 300 pixels wide. Setting WIDTH and HEIGHT to 0 (zero) hides the playback area.

You can also express WIDTH and HEIGHT as percentages of the browser window size. For example, a width of 50% makes the width of the presentation area half the width of the browser window. Keep in mind that different types of media scale with different results. For example, a video scaled larger than its encoded size may not look good. Vector-based clips such as Flash animations, on the other hand, scale more easily to fit different playback areas.

Turning off the Java Virtual Machine

Setting the NOJAVA parameter to true in every <EMBED> tag prevents the browser's Java Virtual Machine (JVM) from starting if it is not yet running:

```
<EMBED SRC="presentation.rpm" WIDTH=300 HEIGHT=134 NOJAVA=true>
```

This parameter primarily affects Netscape Navigator 4, which does not launch its JVM until it's needed. The parameter is recommended because starting the JVM delays presentation playback unnecessarily.

> **Note:** The JVM is required only when you are extending plug-in functionality with Javascript. In this case, omit NOJAVA entirely from the <EMBED> tags. See "Javascript and ActiveX" on page 397 for more information about using Javascript with the <EMBED> tag.

Supporting Other Browsers

To accommodate browsers that do not support the Netscape plug-in, use <NOEMBED> to define a standard hypertext link to your presentation. The unembedded link follows the <EMBED> tag:

```
<EMBED SRC="presentation.rpm" WIDTH=320 HEIGHT=240>
<NOEMBED><A HREF="presentation.ram">Play with RealONE Player.
</A></NOEMBED>
```

In this example, browsers that can play the embedded presentation hide the text between <NOEMBED> and </NOEMBED>. Other browsers ignore the preceding <EMBED> tag and display only the hypertext link. The viewer then clicks the link to play the presentation in RealONE Player.

Adding RealONE Player Controls

With the CONTROLS parameter, you can add RealONE Player controls such as a play/pause button to your Web page. Viewers can then control playback as if they were using RealONE Player as a separate application. For example, the following tag displays the play/pause button in your Web page:

```
<EMBED SRC="presentation.rpm" WIDTH=26 HEIGHT=26 NOJAVA=true
CONTROLS=PlayButton>
```

The following sections describe the embedded RealONE Player controls. You use an <EMBED> tag's WIDTH and HEIGHT parameters to set the control's size. Specifying different pixel sizes other than the suggested values scales the controls larger or smaller. You can also use percentage values for sizes, but this is recommended only for the image window.

For More Information: When adding more than one control to your Web page, see also "Linking Multiple Controls" on page 407.

Tip: Unless noted otherwise, all the controls listed below are compatible with RealPlayer G2, RealPlayer 7, and RealPlayer 8. With those versions of RealPlayer, however, the controls take on a different appearance.

Basic Controls

ImageWindow

The `CONTROLS=ImageWindow` parameter displays a playback window. This control is not required for audio-only presentations. Even if no other controls are visible on the page, the user can typically right-click (on Windows) or hold down the mouse button (on the Macintosh) in the playback area to display a menu of choices such as **Play** and **Stop**. See also "Controlling Image Display" on page 409.

Suggested pixel width: 176 or greater

Suggested pixel height: 132 or greater

All

The CONTROLS=All parameter displays the basic RealONE Player control panel. The control name "default" also works. Functions include play/pause, stop, fast-forward, and rewind. Sliders include a position slider and a volume slider with a mute button that pops up when the speaker button is clicked. Below the buttons are a clip information field, a status panel, a network congestion indicator, and a clip timing field.

Suggested pixel width: 375

Suggested pixel height: 100

Individual Controls and Sliders

ControlPanel

Use CONTROLS=ControlPanel to display a compact RealONE Player control panel. Functions include play/pause, stop, fast-forward and rewind. There's also a position slider, along with a volume slider and mute button that pops up when the speaker button is clicked.

Suggested pixel width: 350

Suggested pixel height: 36

PlayButton (also PlayOnlyButton)

The CONTROLS=PlayButton parameter displays a play button. This turns into a pause button when the presentation plays. If your presentation is accessible to RealPlayers earlier than the RealONE Player, use CONTROLS=PlayOnlyButton instead. In earlier RealPlayers, the PlayButton control includes both play and pause buttons, whereas the PlayOnlyButton control includes just the play button as shown here. Using PlayOnlyButton therefore ensures backwards compatibility.

Suggested pixel width: 36

Suggested pixel height: 26

PauseButton

The CONTROLS=PauseButton parameter displays a pause button. Because the PlayButton control turns into a pause button as a presentation plays, the PauseButton control is generally not necessary with the RealONE Player. To

ensure backwards compatibility with earlier versions of RealPlayer, however, use the `PlayOnlyButton` and the `PauseButton` controls.

Suggested pixel width: 26

Suggested pixel height: 26

StopButton

The `CONTROLS=StopButton` parameter displays a stop button.

Suggested pixel width: 26

Suggested pixel height: 26

FFCtrl

The `CONTROLS=FFCtrl` parameter displays a fast-forward button.

Suggested pixel width: 26

Suggested pixel height: 26

RWCtrl

The `CONTROLS=RWCtrl` parameter displays a rewind button.

Suggested pixel width: 26

Suggested pixel height: 26

MuteCtrl

The `CONTROLS=MuteCtrl` parameter displays a mute button.

Suggested pixel width: 26

Suggested pixel height: 26

MuteVolume

The `CONTROLS=MuteVolume` parameter displays a mute button and volume slider.

Suggested pixel width: 26

Suggested pixel height: 88

VolumeSlider

The `CONTROLS=VolumeSlider` parameter displays a volume slider.

Suggested pixel width: 26

Suggested pixel height: 65

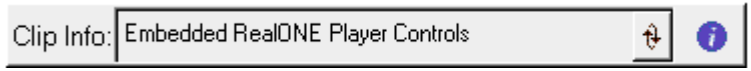
PositionSlider

The `CONTROLS=PositionSlider` parameter displays a clip position slider.

Suggested pixel width: 120

Suggested pixel height: 26

TACCtrl



The `CONTROLS=TACCtrl` parameter displays a clip information field. Presentation information scrolls vertically through this field when the clip first plays. The viewer can redisplay this information by clicking the arrow button. Clicking the "i" button displays the full presentation information in a pop-up window.

Suggested pixel width: 370

Suggested pixel height: 32

> **For More Information:** For instructions on formatting the clip playlist, see "Managing Presentation Information" on page 169.

HomeCtrl

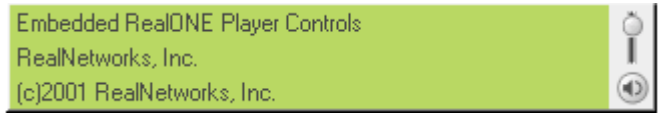
The `CONTROLS=HomeCtrl` parameter displays the RealONE Player logo, which is linked to the RealNetworks Web site. In earlier versions of RealPlayer, this control displays the Real™ logo.

Suggested pixel width: 30

Suggested pixel height: 30

Information Panels

InfoVolumePanel

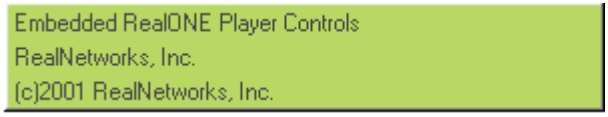


Use CONTROLS=InfoVolumePanel to display presentation information along with the volume slider and mute button. For more on presentation information, see "Defining Information for the SMIL Presentation" on page 169.

Suggested pixel width: 325

Suggested pixel height: 55

InfoPanel



The CONTROLS=InfoPanel parameter displays the presentation information panel. For more on presentation information, see "Defining Information for the SMIL Presentation" on page 169.

Suggested pixel width: 300

Suggested pixel height: 55

Status Panels

StatusBar

The CONTROLS=StatusBar parameter displays the status panel, which shows informational messages. It also includes the network congestion LED and the position field, which shows the clip's current place in the presentation timeline along with the total clip length.

Suggested pixel width: 300

Suggested pixel height: 30

> **Note:** The status bar is included in the All control. If you do not embed a status bar or status field in your page, error messages display in the browser's status bar.

StatusField

The CONTROLS=StatusField parameter displays the message text area of the status bar. If you do not embed a status field or status bar in your page, error messages display in the browser's status bar.

Suggested pixel width: 200

Suggested pixel height: 30

PositionField

The CONTROLS=PositionField parameter displays the position field, which shows the clip's current place in the presentation timeline and the total clip length.

Suggested pixel width: 90

Suggested pixel height: 30

Linking Multiple Controls

The CONSOLE parameter defines a name that unifies <EMBED> tags so that multiple controls work together. For example, you could create three separate <EMBED> tags to define an image window, a play button, and a stop button. By using three tags, you can set the size of each control separately, and define the entire layout with HTML tags. You could put each control in a different HTML table cell, for example.

To tie controls together, define the same CONSOLE name within each <EMBED> tag, or use one of these predefined names:

_master links the control to all other embedded controls on the page.

_unique links the control to no other embedded controls on the page.

You can have multiple console names for separate presentations. For a page showing two video clips, for example, you can define the console names video1 and video2. All controls linked by video1 interoperate, as do all controls linked by video2. But a video1 volume slider, for example, will not affect the volume of a video2 clip.

Tips for Using Consoles

- Every <EMBED> tag must have a SRC attribute. Tags linked by a console name should have the same SRC value.

- If the <EMBED> tags in a console group have different SRC values, the first valid source that RealONE Player finds among those choices becomes the console source. This may not always be the first source listed.
- Clicking a play button for one console stops playback for other consoles. This allows multiple consoles to play separate audio tracks or to use the same image window.

Multiple Controls Example

The following example sets up an image window and two sets of controls (a play button and stop button) for two separate videos, sample1.rm and sample2.rm. The predefined console name _master links the image window to both control sets. The control sets use different console names, however, so they do not link to each other. Clicking each play button therefore starts a different video.

Because each <EMBED> tag must have a SRC value, the image window in the following example uses the same source as the first play button. The viewer simply clicks either play button to start a video. Clicking the other play button stops the first video and plays the second one:

```
<EMBED SRC="presentation.rpm" WIDTH=176 HEIGHT=128 NOJAVA=true
CONTROLS=ImageWindow CONSOLE=_master>

<H4>Video 1</H4>
<EMBED SRC="presentation.rpm" WIDTH=44 HEIGHT=26 NOJAVA=true
CONTROLS=PlayButton CONSOLE=video1>
<EMBED SRC="presentation.rpm" WIDTH=26 HEIGHT=26 NOJAVA=true
CONTROLS=StopButton CONSOLE=video1>

<H4>Video 2</H4>
<EMBED SRC="presentation.rpm" WIDTH=44 HEIGHT=26 NOJAVA=true
CONTROLS=PlayButton CONSOLE=video2>
<EMBED SRC="presentation.rpm" WIDTH=26 HEIGHT=26 NOJAVA=true
CONTROLS=StopButton CONSOLE=video2>
```

Controlling Image Display

The <EMBED> parameters summarized in the following table control aspects of how clips play back in an image window.

Parameters for Image Display

Parameter	Value	Default	Function	Reference
BACKGROUNDCOLOR	*name*|*#RRGGBB*	black	Sets a window color.	page 409
CENTER	true|false	false	Centers the clip.	page 410
MAINTAINASPECT	true|false	false	Determines clip scaling.	page 410
NOLOGO	true|false	false	Suppresses the Real logo.	page 410

The following example shows two of these parameters used in an <EMBED> tag:

```
<EMBED SRC="presentation.rpm" WIDTH=50% HEIGHT=50% NOJAVA=true
BACKGROUNDCOLOR=gray CENTER=true>
```

Setting a Background Color

The BACKGROUNDCOLOR parameter specifies a background color for the image window. The specified background color also shows through if a clip includes transparency. The background color is black by default. You can use an RGB hexadecimal color value (#RRGGBB) or the following color names, shown here with their corresponding RGB values:

white (#FFFFFF)	silver (#C0C0C0)	gray (#808080)	black (#000000)
yellow (#FFFF00)	fuchsia (#FF00FF)	red (#FF0000)	maroon (#800000)
lime (#00FF00)	olive (#808000)	green (#008000)	purple (#800080)
aqua (#00FFFF)	teal (#008080)	blue (#0000FF)	navy (#000080)

Note: SMIL region background colors override this background color. For more on setting SMIL region colors, see "Adding Background Colors" on page 215.

Tip: Appendix D provides background on hexadecimal color values. Note, though, that the <EMBED> tag does not support RGB color values used with SMIL.

Centering a Clip

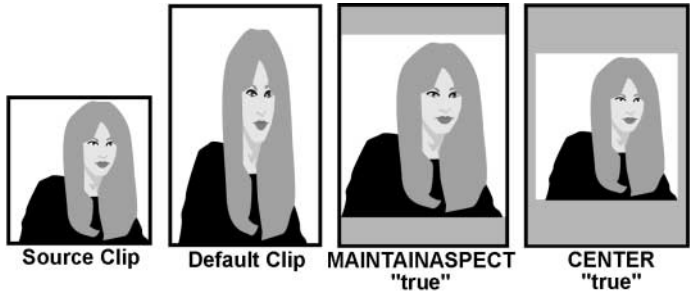
The default value for CENTER is false, which causes the clip to fill the entire playback area. If you set CENTER to true, the clip is centered within the playback area and is displayed at its encoded size. So by setting CENTER to true, you can create a large playback area with WIDTH and HEIGHT and still have the clip play at its normal size. You cannot use CENTER along with MAINTAINASPECT.

Maintaining a Clip's Aspect Ratio

The MAINTAINASPECT parameter, which you cannot combine with CENTER, determines whether the clip's height-to-width ratio stays constant when the clip scales to fit the image window. The default value of false causes this ratio to change as necessary to fill the image window fully. This may distort the source image.

If you set MAINTAINASPECT to true, a clip's height-to-width ratio stays constant. For example, a clip's height-to-width ratio of 1:1 stays constant even if the image window's height-to-width ratio is 2:3. In these cases, the clip is centered in the image window and scaled until one dimension reaches the window's boundaries and the other dimension is within the boundaries. The following illustration shows how clips scale by default, with MAINTAINASPECT set to true, and with CENTER set to true.

Clip Scaling with MAINTAINASPECT and CENTER



Suppressing the Real Logo

When set to true, NOLOGO prevents the Real logo from displaying in the image window before clips play. When there are no clips playing, only the specified background color shows in the window. The parameter is false by default.

Setting Automatic Playback

The parameters listed in the following table can cause a presentation to start playing automatically, and to loop continuously or for a specified number of times.

Parameters for Automatic Playback

Parameter	Value	Default	Function	Reference
AUTOSTART	true|false	false	Starts presentation automatically.	page 411
LOOP	true|false	false	Creates a continuous playback loop.	page 411
NUMLOOP	*integer*	1	Loops playback a set number of times.	page 412
SHUFFLE	true|false	false	Sets shuffle play for a sequence of clips.	page 412

The following example shows two of these parameters in an <EMBED> tag:

```
<EMBED SRC="presentation.rpm" WIDTH=50% HEIGHT=50% NOJAVA=true
AUTOSTART=true LOOP=true>
```

Starting a Presentation Automatically

When set to true, the AUTOSTART parameter starts playback immediately. When you have multiple <EMBED> tags linked by a CONSOLE name, set AUTOSTART to true in just one tag. Leaving AUTOSTART out, or setting its value to false, means that the presentation will not start until the user starts it by clicking an embedded play button, for example.

Looping a Presentation Continuously

If the LOOP parameter is set to true, the presentation continuously loops until the viewer stops it. When you have multiple <EMBED> tags linked by a CONSOLE name, set LOOP to true in just one tag. If you leave LOOP out, the default value of false applies, and the presentation stops after the first playback. The user can play the presentation again by clicking a play button.

> **Note:** The LOOP or NUMLOOP parameters will make an entire SMIL presentation repeat. Within a SMIL file, you can use the repeatDur and repeatCount attributes to repeat individual clips or groups. For more information, see "Repeating an Element" on page 247.

Specifying a Number of Loops

If you specify a parameter such as NUMLOOP=2, the presentation plays the specified number of times and then stops. If you use both LOOP and NUMLOOP, the LOOP parameter is ignored.

Setting Shuffle Play

The SHUFFLE parameter is for use only with Ram or SMIL files that list a single sequence of clips. When set to true, SHUFFLE causes RealONE Player to play the clips in a random order. If you use this parameter with LOOP or NUMLOOP, each loop may use a different playback order.

> **For More Information:** For information on SMIL sequences, see "Playing Clips in Sequence" on page 175. For more on Ram file sequences, see "Writing a Ram File" on page 420.

Laying Out SMIL Presentations

As explained in Chapter 11, you can use a SMIL file to define separate playback regions for different parts of a presentation. This lets you lay out two clips side-by-side, for example. When playing a presentation in a Web page, you can define the layout in SMIL or in HTML.

Defining the Layout with SMIL

To control the layout by using SMIL, you set up the regions and their relative placements in the SMIL file. You then create a Web page playback area large enough to accommodate all SMIL regions. The SMIL file then produces the same layout when played through the Web page or RealONE Player. For example, if your SMIL file creates a playback area 400 pixels wide by 300 pixels high, you define an image window at least as large as this with the <EMBED> tag, as shown in the following example:

```
<EMBED SRC="presentation.rpm" WIDTH=400 HEIGHT=300 NOJAVA=true
CONTROLS=ImageWindow CONSOLE=one>
```

You can then use additional <EMBED> tags linked to the console named one to provide RealONE Player controls for the presentation.

> **Tip:** Typically, your image window is the same size as your SMIL root-layout area, which is described in the section "Root-Layout Area" on page 195.

Defining the Layout with HTML

The second method omits layout information in the SMIL file, defining the layout with HTML instead. You could place an image window and individual RealONE Player controls in separate cells of an HTML table, for example. Each <EMBED> tag then uses a REGION parameter to define a region name. The region each clip plays in is denoted by the region attribute in the SMIL clip source tag:

```
<textstream src="news.rt" region="newsregion"/>
```

Within the HTML page, the <EMBED> tag that plays news.rt would look like this:

```
<EMBED SRC="presentation.rpm" WIDTH=250 HEIGHT=144 NOJAVA=true
CONTROLS=ImageWindow REGION=newsregion CONSOLE=one>
```

You define similar <EMBED> tags to create other regions for other clips listed in the SMIL file. The SRC parameter in each tag lists the same SMIL file. You can also use additional <EMBED> tags linked to the same console to provide RealONE Player controls for the presentation.

> **Tip:** Because you must omit the SMIL layout information when laying out the presentation with HTML, you should not also provide viewers the option to play the same SMIL file directly in RealONE Player. When there is no SMIL layout, RealONE Player automatically creates a layout, but the results may not be what you expect.

CHAPTER

20

PRESENTATION DELIVERY

When you finish building your RealSystem presentation, you place the clips on RealServer or a Web server for delivery to your audience. This chapter explains how to link your Web page to your clips and SMIL files.

Understanding Linking and URLs

Although the process of linking your Web page to your clips is simple, there are two types of mistakes that are easy to make:

1. An incorrect URL can prevent a Web browser or RealONE Player from finding a requested file.
2. An incorrect protocol designation (`http://`, for example) can keep a clip from streaming correctly.

The following sections provide an overview of the process of linking your Web page to streaming clips on a server. The remainder of this chapter then covers the various options for delivering your presentation.

The Ram File

The most common method of linking your Web page to your clips is through a Ram file, which is also called a *metafile*. This file uses the extension `.ram` and often has just one line that gives the full URL to your streaming clip or SMIL presentation. There are two reasons that you need to use a Ram file rather than link your Web page directly to your streaming clips:

1. The Ram file launches RealONE Player.

   The file extension `.ram` causes a Web browser to launch RealONE Player as its helper application. RealONE Player might not launch when you link directly to a clip. When you link your Web page directly to a Flash Player file (extension `.swf`), for example, the browser launches Macromedia's

Flash Player. If you intend to stream your Flash clip, you need to use a Ram file to launch RealONE Player instead.

2. The Ram file provides an RTSP URL for clips on RealServer.

   Clips on RealServer stream over the RTSP protocol, rather than HTTP. This means that the URL used to request the clips must start with `rtsp://` rather than `http://`. Because browsers cannot make RTSP requests, you link your Web page to a Ram file with an HTTP URL. The Ram file then gives RealONE Player the RTSP URL to your clips or SMIL presentation.

   For More Information: See the section "The Difference Between RTSP and HTTP" on page 417.

How a Ram File Works

The following are the steps involved when you link a viewer clicks a hypertext link that connects your Web page to a streaming presentation.

1. The Web browser requests the Ram file from the Web server.
2. The Web server downloads the Ram file to the browser.
3. The Ram file extension (`.ram` or `.rpm`) causes the Web browser to launch RealONE Player.
4. RealONE Player receives the Ram file and requests the clip or SMIL file from the Web server or RealServer.
5. When a SMIL file is used, RealONE Player request the clips based on the URLs in the SMIL file.

The Ram File for Embedded Presentations

For presentations in which RealONE Player pops up as a separate application, you use `.ram` as the Ram file extension. When you embed a clip or presentation in a Web page as described in Chapter 19, however, the Ram file uses the file extension `.rpm`. RealONE Player still plays the presentation, but it does not launch as a separate application. Instead, the browser appears to play the clips. Aside for the file extension, there's no difference between a Ram file for a pop-up presentation (`.ram`), and one for an embedded presentation (`.rpm`).

The Ramgen Alternative to Ram Files

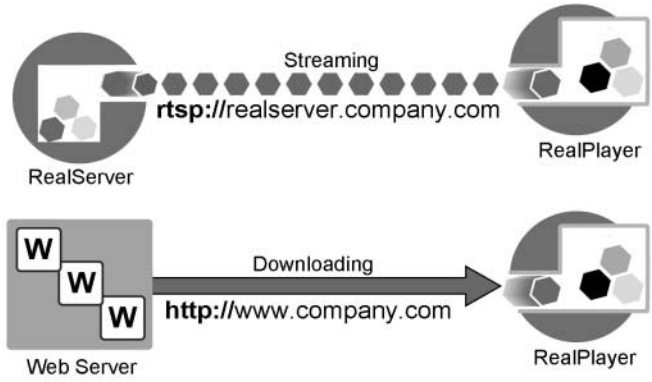
When you stream clips from RealServer, you have the option of using Ramgen, a feature that lets you link your Web page directly to your streaming clips

without using a Ram file. Ramgen uses a specially configured URL that causes the browser to launch RealONE Player and stream clips using RTSP. Although not suited for all streaming presentations, Ramgen can simplify the process of linking your Web to your clips in many cases. For instructions on using Ramgen, see "Using Ramgen for Clips on RealServer" on page 425.

The Difference Between RTSP and HTTP

To deliver HTML pages and graphics, a Web server uses HyperText Transport Protocol (HTTP), as you can see in Web page URLs that begin with `http://`. HTTP downloads files without regard to timelines, making clips with timelines more likely to stall. Although RealServer can also use HTTP, URLs for media clips streamed by RealServer begin with `rtsp://`, which causes RealServer to use RealTime Streaming Protocol (RTSP).

RealServers Stream with RTSP, whereas Web Servers Download with HTTP



Designed specifically for streaming, RTSP enables RealServer to adjust streaming data to keep clips playing smoothly. When two clips play side-by-side, for example, RealONE Player communicates with RealServer about each clip's progress, indicating how much data it needs to keep playback synchronized. RealServer can then adjust the data flow to compensate for changing network conditions, reducing low priority data if necessary to ensure that crucial data gets through. Communication like this is not possible through HTTP.

Which URLs Use Which Protocol

When you assemble a RealSystem presentation, it's important to understand clearly which URLs should use HTTP and which should use RTSP:

- RTSP in SMIL and Ram files for clips on RealServer

  Use rtsp:// in URLs in which RealONE Player requests clips from RealServer. These URLs occur in SMIL files (.smil) and Ram files (.ram or .rpm).

- HTTP in SMIL and Ram files for clips on Web servers

  Use http:// in SMIL and Ram file URLs only if the clips are stored on a Web server instead of on RealServer. Because a Web server does not use RTSP, you cannot use rtsp:// in a URL to a clip stored on a Web server.

- HTTP in Web pages

  Web page links to a Web server or RealServer always start with http://. Web browsers cannot interpret streaming information sent by RealServer through RTSP. The Web browser can connect to RealServer through HTTP, though, because RealServer also uses HTTP.

For More Information: For more on SMIL file URLs, see "Creating Clip Source Tags" on page 145.

Web Server MIME Types

To download a RealSystem presentation from a Web server, the server must be configured with the MIME types listed in the following table. The Web server administrator can configure the MIME types properly.

Web Server MIME Types for RealSystem Files

File Type	Extension	MIME Type
Ram	.ram	audio/x-pn-realaudio
embedded Ram	.rpm	audio/x-pn-realaudio-plugin
SMIL	.smil and .smi	application/smil
RealAudio	.ra	audio/x-pn-realaudio
RealVideo	.rm	application/x-pn-realmedia
Flash	.swf	application/x-shockwave-flash
RealPix	.rp	image/vnd.rn-realpix
RealText	.rt	text/vnd.rn-realtext

Placing Clips on a Server

You typically create your clips and SMIL files on a desktop computer or a workstation, then transfer them to a server, whether RealServer or a Web server, for streaming. The RealServer or Web server administrator can create the content directories for you. When you use RealServer, the RealServer administrator can also set up features such as password authentication and pay-per-view.

Understanding Directory Paths and URLs

It's important to understand that the paths to the clips on a server and the URLs used to request the clips are different. For example, a clip on RealServer may reside in the following path on a Windows machine running RealServer:

```
C:\Program Files\Real\RealServer\Content\video1.rm
```

But the URL used to request the clip may look like this:

```
rtsp://realserver.example.com/video1.rm
```

You'll need the directory path to transfer the clips to the server, and the URL to set up the links for requesting the clip. Your RealServer or Web server can give you the path to the content directories, and tell you the URLs to use to request the clips.

Copying Clips to a Server

When your media clips and SMIL file are ready for streaming, transfer them to RealServer and place them in the directories prepared by the administrator. If the server is on the same local area network (LAN) as your computer, you can often just copy the files to the server over the network. Otherwise, you can usually transfer files to a server over the Internet using FTP (file transfer protocol).

The FTP protocol is designed to copy files from one computer to another. Many computers have an FTP application preinstalled. FTP applications are also available for download from many Internet software archives. Your RealServer or Web server administrator will have to set up FTP access to the server machine for you, as well as give you an FTP user name and password.

Note that FTP distinguishes between a "text" mode for transferring text-only files, and a "binary" mode for transferring non-text files such as images, videos, audio clips, and animation clips. Some FTP programs can set this

mode automatically. With other programs, though, you must set the transfer mode yourself. If you transfer a non-text file in text mode, the file may become corrupt and fail to stream. Fortunately, FTP never modifies the original file on your computer, so you can simply transfer the file again using the binary mode.

> **Tip:** RealProducer and RealSlideshow can transfer files to a server automatically. Refer to their manuals or online help for more information.

Launching RealONE Player with a Ram File

A Ram file is a text file with the extension `.ram` (`.rpm` for playback in a Web page). When a browser receives this file, it launches RealONE Player as a helper application. RealONE Player then requests the clips listed in the Ram file. As described in "Using Ramgen for Clips on RealServer" on page 425, RealServer can launch RealONE Player without using a Ram file. When doing any of the following, however, you may need to write a Ram file:

- Streaming from a RealServer not set up to use Ramgen.
- Hosting a RealSystem presentation on a Web server.
- Playing back clips that reside on the viewer's local computer.
- Opening a clip in a specific RealONE Player mode, such as full-screen mode.

Writing a Ram File

A basic Ram file has only one line: the full URL to a clip or SMIL file. A Ram file can also list multiple URLs to different clips, each URL on a separate line. This causes RealONE Player to play those clips in sequence. With RealONE Player, a Ram file can also contain multiple URLs to SMIL files and even to other Ram files.

> **Warning!** RealPlayer 8 and earlier cannot play Ram files that list other Ram files, or more than one SMIL file.

➤ To write a Ram file:

1. Open any editor or word processor that can save files as plain text. On the top line, enter the full URL of the SMIL file or the media clip. Add the full URL to each subsequent clip or file on a new line. You may need to verify

URLs with your RealServer or Web server administrator. The following shows sample URLs to a SMIL file, depending on whether the file resides on RealServer, a Web server, or the viewer's local computer:

RealServer	`rtsp://realserver.example.com/sample1.smil`
Web server	`http://www.example.com/sample1.smil`
Local	`file://sample1.smil`

> **For More Information:** For more information on RTSP URLs, see "Linking to Clips on RealServer" on page 153.

2. For a presentation played back from RealServer, you can support earlier versions of RealPlayer (such as RealPlayer 4 or 5) just as Ramgen does with the `altplay` option. To do this, add the marker `--stop--` after the RTSP URL and then specify the URL for the older clip just as it appeared in your previous Ram file. Here's an example:

   ```
   rtsp://realserver.example.com:554/media/sample.smil
   --stop--
   pnm://realserver.example.com:7070/media/old_sample.rm
   ```

   The second URL specifies the older RealSystem protocol with `pnm://` and designates RealServer's PNA port. When RealONE Player or RealPlayer connects, it chooses the URL based on its favored protocol. For this reason, you cannot list two URLS that both use the same protocol, whether `rtsp://`, `pnm://`, or `http://`.

3. Save the Ram file as plain text with a `.ram` extension (played in RealONE Player) or a `.rpm` extension (played in a Web browser).

4. Move your Ram file to RealServer or your Web server. Even if all your media clips are on RealServer, you can place the Ram file on your Web server. When the browser receives a Ram file, it turns it over to RealONE Player, which uses the URLs in the file to request clips. Hence the Ram file and the media clips do not need to reside on the same computer.

5. For `.ram` files, link your Web page to the Ram file by using an HTML hyperlink such as this:

   ```
   <a href="http://www.example.com/media/sample.ram">click for video</a>
   ```

   For `.rpm` files, incorporate the link URL in the `<EMBED>` tag as described in "Using <EMBED> Tags" on page 398. If the Ram file is on RealServer, the URL must not use the `ramgen` parameter.

Setting a Presentation's Starting Mode

In the Ram file, you can control how RealONE Player initially displays a clip or SMIL presentation. You can play a clip at double its normal size, for example, or open the RealONE Player in its toolbar mode. To set the starting mode, add one of the options listed in the following table to the end of the Ram file URL.

Options for Setting the Initial Playback Mode

Option	Function
`?screensize="double"`	Opens the clip or presentation at double its normal size.
`?screensize="full"`	Starts the clip or presentation in full-screen mode. This hides all windows to make the viewer's monitor look like a television screen.
`?screensize="original"`	Opens the clip or presentation at its normal size. This is the default behavior if you omit `screensize`.
`?mode="normal"`	Opens RealONE Player in its normal mode, in which controls are grouped around the playback window.
`?mode="toolbar"`	Opens RealONE Player in toolbar mode, in which the controls appear at the bottom of the screen.
`?mode="theater"`	Opens RealONE Player in theater mode, in which the controls are put in toolbar mode, and the media presentation appears centered on a darkened screen.

The following example opens a SMIL presentation in full-screen mode:

```
rtsp://realserver.example.com/media/sample1.smil?screensize="full"
```

The next example opens a RealVideo clip at double its normal size:

```
rtsp://realserver.example.com/media/video1.rm?screensize="double"
```

To include two options in the URL, use a question mark (?) before the first option, and then separate the second option with an ampersand (&). The following example opens a RealVideo clip at double its normal size and sets RealONE Player to its toolbar mode:

```
rtsp://realserver.example.com/media/video1.rm?screensize="double"&mode="toolbar"
```

Tips for Setting a Starting Size

- You do not need to specify a starting mode when using `screensize="full"`.

- RealONE Player may not offer full-screen mode on all operating systems. If RealONE Player for a given operating system does not offer full-screen mode, it plays the presentation at its normal size.
- If RealONE Player offers full-screen mode but has not yet played a clip full-screen, it may first perform a test of this playback mode.
- The double and full-screen modes work best for high-speed clips. They are not recommended for presentations delivered through modems.
- Always test playback when using double and full-screen modes to ensure that the visual quality is acceptable. Some types of clips may not scale well.
- In full-screen mode, the viewer can control RealONE Player through a context menu accessed by right-clicking (on Windows) or holding down the mouse button (on Macintosh).
- Do not use these options when embedding a presentation in a Web page through a `.rpm` file.

Adding Comments to a Ram File

You can add a comment to a Ram file by using a pound sign (#) as the first character on a line. The following example shows two lines commented out of a Ram file:

```
# Two videos and a SMIL presentation
# streamed from RealServer.
rtsp://realserver.example.com/media/video1.rm
rtsp://realserver.example.com/media/video2.rm
rtsp://realserver.example.com/media/sample2.smil
```

Examples of Linking a Web Page to Clips

The following sections provide some examples of linking a Web page to clips or a SMIL presentation that resides on RealServer or a Web server.

Linking to a Single Clip

Suppose you have a single RealVideo clip called `video1.rm`. You can simply link your Web page to a Ram file (`play_video1.ram`) that resides in the same directory as the Web page:

```
<a href= "play_video1.ram">Play the video!</a>
```

The Ram file then gives RealONE Player either the full RTSP URL to the clip on RealServer:

```
rtsp://realserver.example.com/video1.rm
```

or the full HTTP URL to the clip on a Web server:

```
http://www.example.com/video1.rm
```

Linking to an Embedded Clip

Suppose that you've embedded a RealVideo clip called `video1.rm` in your Web page according to the instructions in Chapter 19. You can link to a Ram file (`play_video1.rpm`) that resides in the same directory as the Web page within the `<EMBED>` tag:

```
<EMBED SRC="play_video1.rpm" WIDTH=300 HEIGHT=134>
```

The Ram file then gives RealONE Player either the RTSP URL to the clip on RealServer:

```
rtsp://realserver.example.com/video1.rm
```

or the HTTP URL to the clip on a Web server:

```
http://www.example.com/video1.rm
```

Linking to a SMIL Presentation

Linking to a SMIL file is similar to linking to a clip. However, because a SMIL file contains the URLs to clips in the presentation, the SMIL file itself can reside on any server. Suppose you have a SMIL file named `presentation.smil`. You can simply link your Web page to a Ram file (`play_presentation.ram`) that resides in the same directory as the Web page:

```
<a href= "play_presentation.ram">Play the video!</a>
```

The Ram file should give RealONE Player the full RTSP URL to the SMIL file if it resides on RealServer:

```
rtsp://realserver.example.com/presentation.smil
```

or the HTTP URL to the file if it resides on a Web server:

```
http://www.example.com/presentation.smil
```

The SMIL file itself should contain the full URLs to clips in its source tags, as in the following example:

```
<video src="rtsp://realserver.example.com/video1.rm" .../>
```

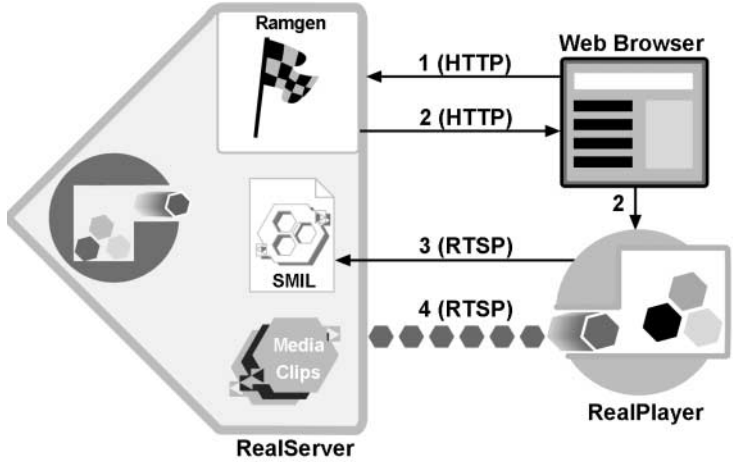
For More Information: See "Writing Clip Source URLs" on page 151 for more information on SMIL file URLs.

Using Ramgen for Clips on RealServer

With RealServer, you can use Ramgen to launch RealONE Player automatically, eliminating the need to write a separate Ram file. Your Web page URL simply points to your media clip or SMIL file on RealServer and includes a `ramgen` parameter. If your RealServer does not use Ramgen, you can write a Ram file as explained in "Launching RealONE Player with a Ram File" on page 420. A Ram file also enables you to use some RealONE Player features, such as playing a clip at double or full-screen size.

The following illustration shows the process of requesting a presentation through Ramgen. This example uses a SMIL file that coordinates multiple clips, but you can also link to a single clip directly without using SMIL.

Requesting a Presentation from RealServer Using Ramgen



1. Using HTTP, the Web browser requests the SMIL file from RealServer. The URL includes a parameter that invokes Ramgen.
2. RealServer's response causes the Web browser to launch RealONE Player as a helper application and to give it the URL to the SMIL file.
3. RealONE Player requests the SMIL file from RealServer using RTSP.
4. With the information in the SMIL file, RealONE Player requests and receives the streaming media clips.

Linking Your Web Page to RealServer Using Ramgen

With your clips on RealServer, link your Web page to the SMIL file by using an HTML hypertext link that looks like the following:

```
<a href="http://realserver.example.com:8080/ramgen/media/sample.smil">...</a>
```

If the presentation plays back directly in the Web page, the URL occurs within an <EMBED> tag and looks like this:

```
SRC="http://realserver.example.com:8080/ramgen/media/sample.smil?embed"
```

In these examples, the /ramgen/ parameter causes the Web browser to launch RealONE Player without the use of a separate Ram file. This parameter designates a virtual directory in RealServer, and can be followed in the URL by actual directory listings. The following table describes the components of these URLs. Contact your RealServer administrator to get the actual RealServer address, HTTP port, and directory structure.

URL Components in a Web Page Link to RealServer

URL Component	Function
http://	This causes the browser to contact RealServer through HTTP. (Web browsers do not use RTSP.)
realserver.example.com	This address varies for each RealServer. It typically uses an identifier such as realserver instead of www. It may also use a numeric TCP/IP address, such as 204.71.154.5.
:8080	This is the port RealServer uses for HTTP connections. Separate the port and address with a colon. You can leave the port number out if RealServer uses port 80 for HTTP connections. Include the port number if RealServer uses any port other than 80 for HTTP.
/ramgen/	This parameter launches RealONE Player without the use of a separate Ram file.
/media/	Following /ramgen/, the URL may list other directories, depending on where the clip resides on RealServer.
sample.smil	This is the SMIL file for your presentation. If you have only one clip to stream, you can link directly to that clip instead of to a SMIL file.

(Table Page 1 of 2)

URL Components in a Web Page Link to RealServer (continued)

URL Component	Function
?altplay=*file.ext*	This Ramgen option specifies an alternate presentation created for earlier versions of RealPlayer. See "Listing Alternative Presentations with Ramgen" on page 427.
?embed	This Ramgen option embeds the presentation in a Web page. See Chapter 19 for complete information on Web page playback.

(Table Page 2 of 2)

Listing Alternative Presentations with Ramgen

With altplay, you can use a single link to stream new clips to RealONE Player, while streaming older clips to earlier versions of RealPlayer. Suppose that you have a RealVideo 5 clip and a RealVideo 8 clip laid out using SMIL. You link to the SMIL file using Ramgen as described in the preceding section, and you include altplay to list the older clip:

```
<a href="http://.../ramgen/media/sample.smil?altplay=old_sample.rm">
```

This link instructs RealServer to point RealPlayer G2 or later to sample.smil. Earlier versions of RealPlayer receive the URL to old_sample.rm. RealServer uses the streaming protocol appropriate for each RealPlayer version, whether RTSP or the older PNA. Note that altplay specifies the clip, not a Ram file. Because of this, the older clip must reside in the same directory as the new content.

Combining Ramgen Options

The question mark operator (?) separates Ramgen options from the main URL. To use multiple Ramgen options, you use a question mark before the first option and separate the remaining options with ampersands (&). The order of options does not matter. For example, the following link uses altplay and embed:

```
<a href="http://.../ramgen/media/sample.smil?embed&altplay=old_sample.rm">
```

You can use the ? operator to include earlier Ram file options when using altplay. If your Ram file URL for a RealVideo 5 clip specified an end time, for example, include that option in the Ramgen URL after altplay. The following example shows an end time set for old_sample.rm:

```
<a href="http://.../ramgen/media/sample.smil?altplay=old_sample.rm&end=7:45">
```

Limitations on Web Server Playback

If you do not have access to RealServer, you can host your presentation on a Web server. Although not as robust as RealServer streaming, Web server playback provides a reasonable method for sending simple presentations to a small number of viewers. Because Web servers are not designed to manage bandwidth or keep multiple clips synchronized, presentations delivered by a Web server are more likely to stall than when streamed by RealServer. To ensure that a presentation hosted by a Web server plays as smoothly as possible, observe the following points.

No SureStream Clips Encoded for Multiple Bandwidths

A Web server cannot send just one stream from a SureStream clip encoded for several bandwidths. Instead, it downloads the entire clip, causing a very high preroll. You must therefore encode each RealAudio or RealVideo clip for just one bandwidth. When using RealProducer, select the option for Web server playback and choose your target audience. To support multiple bandwidths, encode separate clips for various bandwidths and use SMIL to let RealONE Player choose which clip to play.

> **For More Information:** For more on using SMIL to list clip choices, see "Switching Between Bandwidth Choices" on page 370.

No Secure RealAudio and RealVideo Clips

When you encode RealAudio and RealVideo clips with RealProducer, you have an option to prevent RealONE Player users from recording the streamed clips to their computers. This feature works only when RealServer streams the clips. When a Web server delivers the clips, users still cannot record the clips through RealONE Player, but their Web browsers will cache the clips. Additionally, any user can click on your Web page hypertext links and use **Save as...** commands to download the clips from the Web server.

Limited Ability to Keep Parallel Clips Synchronized

A Web server does not consider clip timelines when downloading data. Nor does it receive feedback from RealONE Player about the presentation's progress. Web server playback therefore makes it harder for RealONE Player to

keep clips synchronized. A presentation that plays large clips in parallel may stall when the RealONE Player connection has little bandwidth to spare.

No Way to Set Image Streaming Speeds

As the section "Setting a Clip's Streaming Speed" on page 146 explains, you can set an image clip's streaming speed with a `<param/>` tag when you use RealServer. This SMIL attribute has no effect on presentations delivered with a Web server, however. A Web server will download the image as quickly as possible, which may interfere with other clips displaying at the same time.

RealPix Presentations Require Clip Size Information

RealServer determines when to stream each RealPix image based on the image's place in the presentation timeline. Because a Web server cannot do this, you must indicate each image's file size in the RealPix markup. This enables RealONE Player to calculate when to request an image from the Web server so that all image data has arrived by the time the image displays. If the file size information is missing, RealONE Player requests all images when the presentation starts, causing a high preroll.

SMIL File Optional

When delivering a single clip or a few clips played in sequence, you do not need a SMIL file. Instead, you can simply list the clips in order when writing your Ram file, as described in "Launching RealONE Player with a Ram File" on page 420. However, you can also have your Ram file specify a SMIL file that lists the clip locations, creates a layout, times the presentation, and so on.

> **Note:** RealNetworks does not recommend using long or complex SMIL files when delivering presentations with a Web server. Limit your SMIL file to a few clips played in sequence or in parallel.

SMIL Internal Timing Commands Do Not Work

Although you can use SMIL to lay out and time your presentation, you should not use the `clipBegin` and `clipEnd` attributes. A Web server cannot begin to download a clip at a certain point in its timeline. With `clipBegin="5min"`, for example, RealONE Player must wait until it has received the first 5 minutes of clip data before it can play the clip. This results in an unacceptably long wait.

For More Information: "Setting Internal Clip Begin and End Times" on page 242 describes these SMIL commands.

No Presentation Seeking

Because a Web server cannot jump to a new position in a clip's timeline, the RealONE Player position slider cannot fast-forward the clip. If the viewer moves the slider forward, playback pauses as the clip continues to download at its normal rate. RealONE Player resumes playback once the clip data reaches the specified timeline position.

No RTSP URLs

Because Web servers do not support RTSP, all URLs in presentations hosted by Web servers should begin with `http://`. This includes all URLs in a SMIL file or Ram file.

No Live Broadcasting

Live broadcasting is not possible because Web servers can download only clips that are stored on disk.

Testing Your Presentation

Use the following guidelines to make sure your presentation works well and reaches its target audience:

- Test your presentation in "real world" conditions. If you target 56 Kbps connections, for example, request the presentation over a 56 Kbps modem.
- Check that the presentation has a preroll (initial buffering) under 15 seconds. After preroll, the presentation should not rebuffer under normal network conditions.

  For More Information: See "Buffering" on page 31.

- Verify that video and audio quality is acceptable.
- For a multiclip presentation, verify that clips stay synchronized. Ensure that no stalling occurs because of too many clips playing at the same time or a single clip requiring too much bandwidth. Make sure that clips

introduced during a presentation in progress do not stall playback by requiring too much buffering when they start.

- Make sure that your presentation works well for an "average" CPU for your audience. For general Web delivery, test playback on both Pentium and Power Macintosh computers with clock speeds around 300 MHz.

  Tip: If your presentation is CPU-intensive because it uses complex Flash animation or high-bandwidth video, for example, note this in your Web page.

- When streaming RealAudio clips, ensure that sound quality is acceptable. You may need to experiment with RealAudio codecs to find the best balance between clip bandwidth use and sound quality.
- Test all hypertext links and interactive functions.
- When embedding a presentation in a Web page, verify that the playback window has the correct location and controls.

Advertising on RealGuide

Every day, thousands of people visit RealGuide, RealNetworks' online guide for streaming media sites and live events (**http://realguide.real.com**). If you regularly host streaming media presentations of interest to the public, or if you have a live event you want to advertise, you can submit your listing to RealNetworks. Simply complete the following online form to list your site or live event:

- **http://realguide.real.com/info/?page=submit**

In the form, you provide the site or event name, the URL, a short description, and the name of a contact person. A RealGuide staff member then verifies your site or event before including it on RealGuide. (RealNetworks reserves the right to edit or refuse submissions.) For live events, please submit your request at least one business day in advance. If you have questions or need to change a listing, please contact RealGuide from this Web page:

- **http://realguide.real.com/info/?page=fbform**

Using RealNetworks Logos

When you create RealSystem content, RealNetworks encourages you to add RealSystem logos to your Web page. You can provide a RealONE Player download link button, for example, so that users can get RealONE Player from RealNetworks' Web site and view your content. You can read RealNetworks' trademark policies and get RealSystem and RealONE Player logos at the following address:

http://www.realnetworks.com/company/guide/index.html

PART

VIII

APPENDIXES

Whether you're a novice or a professional, these appendixes will help you as you build your presentation. Appendix A takes up basic questions beginning users often ask. Once you become more familiar with RealSystem, Appendix B will guide you to areas of this manual that address specific production issues. For the advanced user, Appendix C and Appendix D provide quick references to SMIL tags and color values, respectively. Appendix E does the same for RealText markup. Appendix F explains common file extensions, while Appendix G lists the RealONE Player language codes.

APPENDIX

A

BASIC QUESTIONS

This appendix, provided for the beginning RealSystem user, answers often-asked questions about producing streaming media clips for RealSystem. It also provides URLs for Web sites where you can find tools and helpful information about developing streaming media presentations.

Creating Streaming Clips

RealProducer is the basic tool you use to create clips. Both the *RealProducer User's Guide* and the product's online help guide you through the encoding process. This production guide provides background information and tips on creating high-quality streaming media.

How do I make streaming audio and video clips?

You start with an audio or video source file in a digitized format on your computer. You then use RealProducer's encoding wizard to select the file and set encoding options. The encoding process creates a new streaming clip, leaving the source file unchanged.

Can I encode RealVideo directly from a video camera?

Yes. RealProducer accepts live video input from a camera and live audio input from a microphone. The camera and microphone connect to an audio/video capture card on your computer. RealProducer's encoding wizard then lets you select the live input as the source. In this case, you go directly from live input to encoded clip without creating a digitized source file.

How do I ensure the best quality for streaming clips?

Quality starts at the source. You need high-quality video and audio input for RealProducer to create high-quality streaming clips. Chapter 3 and Chapter 4 include tips on producing good audio and video, respectively. If you are new to

media production, learn your editing hardware and software thoroughly, paying close attention to the manufacturers' recommendations for producing high-quality media files.

What other clips can I stream?

In addition to audio and video, RealSystem can stream the following types of clips:

- Macromedia Flash animation
- GIF, JPEG, and PNG images
- RealPix clips for streaming slideshows
- RealText clips for streaming text

Getting Production Tools

To produce streaming media clips, you need audio and video production tools as well as RealProducer to handle the encoding.

What audio and video editing tools can I use?

You can use any hardware or software designed for capturing and editing audio or video. The digitized output must be in a format that RealProducer accepts, however. Some video editing programs save digitized video in a proprietary format that RealProducer cannot read. However, these programs typically let you export the video to a common format that RealProducer accepts, such as AVI, QuickTime, or MPEG.

> **Tip:** You can purchase hardware and software for capturing and editing audio or video from RealStore at **http://www.realstore.com**.

What digitized audio and video formats does RealProducer accept as input?

RealProducer accepts many common audio and video formats. These may vary by operating system, though. RealProducer on Macintosh accepts the formats widely used on the Macintosh, such as QuickTime, whereas RealProducer on Windows or Unix supports the formats widely used on those operating systems. Check the RealProducer manual for your operating system for a list of accepted formats. Information is also available at the following Web page:

http://www.realnetworks.com/products/producer/freevsplus.html

Where can I get RealProducer?

RealNetworks makes versions of RealProducer for Windows 95/98/NT/2000, Macintosh, and Linux. You can download the free version or purchase RealProducer Plus at RealNetworks' Web site:

http://www.realnetworks.com/products/index.html

How do I create a streaming slideshow from still images?

Using RealSlideshow's graphical interface, you can create streaming RealPix presentations from still images. You can even add a soundtrack, or record a narration for each image. You can download RealSlideshow from this Web address:

http://www.realnetworks.com/products/index.html

You can also create RealPix presentations by hand with the RealPix markup language, which is described in *RealPix Authoring Guide,* available at this Web page:

http://service.real.com/help/library/encoders.html

How do I create streaming Flash animation?

You create animation with Macromedia Flash. You can develop animations with Flash 2, 3, or 4. Chapter 5 provides tips for making Flash animation stream well with RealSystem. It doesn't explain how to create Flash animations, however. You can learn more about Flash at from Macromedia's Web site:

http://www.macromedia.com/software/flash/

What's the RealSystem Authoring Kit?

The RealSystem Authoring Kit bundles several RealNetworks tools and manuals into one archive that you can download. This gives you one source for the basic tools and information you need to create streaming media clips. The Authoring Kit is available free at:

http://www.realnetworks.com/products/authkit/

Using SureStream

SureStream provides advanced streaming technology for RealSystem. For more information about SureStream, read "SureStream RealAudio and RealVideo" on page 35.

What is SureStream?

SureStream is a technology that lets a single RealAudio or RealVideo clip stream at different bit rates. It does this by bundling into a single clip multiple streams, each of which runs at a different bit rate. You can make a SureStream clip that streams at either 28.8 Kbps or 56 Kbps, for example. When users request the clip, they automatically receive the stream that best matches their RealONE Player connection speed.

How do I make a SureStream clip?

Using RealProducer, you can choose to use SureStream when you encode audio or video input. The number of SureStream streams you can encode in the clip depends on the type of RealProducer you use. RealProducer Basic encodes two speeds per clip, whereas RealProducer Plus encodes up to eight speeds per clip.

Can I use SureStream with a Web server?

No. A SureStream clip has several streams encoded in a single clip. Unlike RealServer, a Web server cannot extract a specific stream to send to RealONE Player. If you plan to deliver clips from a Web server, you need to set RealProducer to use single-rate encoding.

Writing SMIL Files

Chapter 7 explains the basics of SMIL. Appendix B explains how to do some common tasks with SMIL. Appendix C provides a SMIL reference you can use once you are comfortable with SMIL.

What is SMIL?

Pronounced "smile," SMIL stands for "Synchronized Multimedia Integration Language." It is an industry-standard markup language used to lay out and time streaming media presentations. SMIL works for RealONE Player the way HTML works for a Web browser.

Is it necessary to use SMIL?

Not always. When you want to stream just one clip, such as a single RealVideo clip, you don't need to use SMIL. You just link your Web page to the clip through a Ram file. For more information, see "What is a Ram file?" on page 440.

When should I use SMIL?

When you stream multiple clips, SMIL gives you the means to lay out the presentation and time its clips. It also provides other features, such as letting you create hyperlinks that start a new presentation. For a rundown of basic SMIL features, see "Understanding SMIL" on page 127.

How do I write SMIL?

SMIL is a simple markup language that you can write with a word processor or text editor. Some software tools (RealSlideshow, for example) create SMIL files automatically.

Streaming Clips

You can stream clips yourself with RealServer, through a service provider that has RealServer available, or, in some cases, from a Web server.

Do I need to have RealServer in addition to RealProducer?

Not necessarily. RealServer streams the clips created by RealProducer. To run RealServer, you need a computer connected to an intranet or one that has a direct presence on the Internet. You cannot run RealServer if you use an Internet service provider (ISP) to connect to the Internet. If you use an ISP, check whether they have RealServer and whether they can host your streaming presentations for you.

What operating systems does RealServer run on?

RealServer runs on Windows NT/2000 and many Unix platforms, including Linux. For a list of available platforms, visit RealNetworks' technical support Web site at **http://service.real.com**.

Where do I get RealServer?

RealServer is available on the RealNetworks Web site at **http://www.realnetworks.com/products/index.html**. RealServer Basic is free.

Can I stream clips from a Web server instead of RealServer?

Sometimes. A Web server can deliver many types of clips, including RealAudio and RealVideo. There are limits to Web server delivery, however. If you plan to use a Web server for clip delivery, read "Limitations on Web Server Playback" on page 428 first.

What is a Ram file?

A Ram file, also called a "metafile," is a simple text file with the extension `.ram` that typically consists of just one line: the URL to a streaming presentation. Your Web page does not link directly to your presentation. Instead, it links to the Ram file, which ensures that RealONE Player launches. RealONE Player then uses the URL in the Ram file to request the presentation. "Launching RealONE Player with a Ram File" on page 420 explains how to write a Ram file.

> **Tip:** When you stream clips with RealServer, you can eliminate the Ram file by using the Ramgen utility. For more information, see "Using Ramgen for Clips on RealServer" on page 425.

If I use SMIL, do I need a Ram file?

Yes. The SMIL file lists the URLs for clips. The Ram file supplies RealONE Player with the URL to the SMIL file (or to your streaming clip, if you're not using SMIL). The Ram file is always necessary because its `.ram` extension launches RealONE Player.

Can I place clips in a Web page?

Yes. Chapter 19 explains how to embed clips and RealONE Player controls in a Web page. RealONE Player still plays the clips, but it does so "behind the scenes" as a browser plug-in rather than by launching separately.

Why does RealServer use RTSP rather than HTTP?

Web servers use HTTP to deliver Web pages and graphics. HTTP is designed to download small files quickly and efficiently. It is not suited for streaming large

media clips, though. RTSP, which stands for "RealTime Streaming Protocol," is an industry-standard protocol that overcomes the deficiencies of HTTP for streaming media. RTSP enables RealServer and RealONE Player to stream long clips and compensate for changing network conditions.

How do I stream clips with RTSP?

When a clip resides on RealServer, make sure that the URL used to request it starts with `rtsp://` rather than `http://`. An RTSP URL must be in a file read by RealONE Player, such as a Ram file or a SMIL file. It cannot be in an HTML hyperlink, because a Web browser does not know how to make an RTSP request. For more on this, see "The Difference Between RTSP and HTTP" on page 417.

Advertising

The RealServer administrator performs most of the work for setting up ad streaming, as described in *RealServer Administration Guide*.

How does advertising with RealSystem work?

Ad streaming uses `<RealAdInsert/>` tags in SMIL files to designate when and where ads appear in presentations. When RealServer serves a SMIL file, it replaces the tags with URLs to ad files. These URLs come from a separate ad server, and the RealServer administrator can configure RealServer to work with virtually any ad-serving system.

What kinds of ads can I stream?

RealServer can stream banner ads in image formats such as GIF, JPEG, and PNG. It can display one ad per presentation, or it can make new ads appear at specific intervals during the presentation. RealServer can also stream ads in RealAudio, RealVideo, and Flash formats. You can have ads either precede the requested clips or appear as commercial breaks during the clips.

How do I set up advertising?

You need RealServer with the optional advertising extension installed. You also need either to have an ad server, or to sign up with an online ad provider such as DoubleClick. The RealServer administrator configures RealServer to work with your ad-serving system, determining what types of ads are to be

streamed. You then write SMIL files with `<RealAdInsert/>` tags that specify where ads are to be placed.

Do I have to write SMIL files to stream ads?

No. Writing your own SMIL files gives you a more flexible way to stream ads, but RealServer can automatically include ads with requested clips or SMIL presentations. If your RealServer hosts a large number of RealAudio or RealVideo clips, for example, you can simply let RealServer lay out an ad for each clip.

Broadcasting

For full information about broadcasting media, see *RealProducer User's Guide* and *RealServer Administration Guide*.

What do I need for broadcasting over a network?

You need the following:

- An audio or video capture card on your computer, to digitize the input from a microphone or camera.
- RealProducer on the same computer as the capture card, to encode the output in a streaming format and send the stream to RealServer.
- RealServer, to broadcast the stream to one or more RealONE Players. RealServer typically does not run on the same computer as RealProducer.

Can I broadcast through my ISP?

Possibly. If you connect to the Internet through an ISP, you may be able to broadcast streaming media, provided that your ISP has RealServer available and offers broadcasting services. To do this, you will need a fast Internet connection to your ISP. You cannot broadcast through an ISP by running RealServer on your home computer.

Can I use SureStream in a broadcast?

Yes. Using SureStream is recommended because it ensures that users connecting at different speeds will each receive the best possible stream. You need to make sure, however, that the computer running RealProducer has enough power to encode all the SureStream streams at the same time. Check

RealProducer's manual or online help for system requirements, and perform a trial run before streaming the actual broadcast.

Can I broadcast with a Web server instead of RealServer?

No. You need RealServer to broadcast streaming presentations. Web servers are designed to serve HTML pages and graphics to different users at different times. They are not designed to broadcast the same presentation to multiple users simultaneously.

Does a broadcast have to be live?

No. "Broadcasting" means to send out a stream that more than one RealONE Player user can view at the same time. The broadcast can be live, meaning that the input originates from a microphone or video camera. Or it can be prerecorded, meaning that it originates from a digitized clip prepared in advance. If it's prerecorded, you don't need to use RealProducer during the broadcast. You just put the clip on RealServer and then set up RealServer to broadcast the clip as a simulated live event.

Can I use SMIL with a broadcast?

Yes. You can use SMIL to include ads with the broadcast, or deliver static clips alongside the broadcast. In the SMIL file, you simply treat the broadcast as a static clip. The only difference is that you use a special URL created by the RealServer administrator that identifies the resource as a broadcast rather than a clip.

How many people can I reach with a broadcast?

That depends entirely on your RealServer and the network bandwidth it has available. Each RealServer can broadcast to a specific number of RealONE Players, as defined in its license agreement. For large broadcasts, you can use a network of RealServers to reach thousands of RealONE Players.

Can RealNetworks broadcast clips for me?

Yes. Real Broadcast Network (RBN) offers a wide range of services for hosting broadcasts. Learn more about RBN at:

http://www.realnetworks.com/rbn/index.html

Getting Technical Support

RealNetworks offers a range of technical support features and documentation.

How do I get technical support from RealNetworks?

RealNetworks Technical Support operates an extensive Web site at **http://service.real.com**. The site includes answers to frequently asked questions, a documentation library, and a searchable knowledge base. To place a service call with Technical Support, fill out the e-mail form at the following Web page:

http://customerrelations.real.com/scripts/rnforms/contact_tech_service.asp

Where can I find additional documentation?

RealNetworks Technical Support maintains a documentation library at **http://service.real.com/help/library/index.html**. Most documents are available as bundled HTML archives that you can download, uncompress, and read with a Web browser. Many documents are also available in PDF format, which is suitable for printing. To read PDF files, you need Adobe's Acrobat Reader, which is available from Adobe's Web site:

http://www.adobe.com/products/acrobat/readstep.html

Where should I go for the latest RealSystem information?

The RealNetworks DevZone is the main information site for content authors and software developers working with RealSystem products. You can find it at the following Web address:

http://www.realnetworks.com/devzone

APPENDIX

B

PRODUCTION TASKS

Intended for the intermediate user, this appendix addresses specific production questions by referring you to the appropriate section in this manual. If you have questions about a specific SMIL tag or attribute, you may find Appendix C more useful.

Basic SMIL Issues

Chapter 7 explains the basics of using SMIL 2.0 in a RealSystem presentation.

Basic SMIL 2.0 Issues

Question	Answer
Why should I use SMIL?	page 127
What versions of RealONE Player can play SMIL 2.0 presentations?	page 129
Besides more features, are there differences between SMIL 1.0 and 2.0?	page 142
How do I update my SMIL 1.0 presentation to SMIL 2.0?	page 143
Where can I get the SMIL 2.0 specification?	page 127
Will my presentation work with other SMIL-based players?	page 132
How do I write a SMIL file?	page 133
Is a closing slash always necessary with a SMIL tag?	page 137
How do I add comments to a SMIL file?	page 137
What values can I use for the ID in a SMIL tag?	page 138
What is the rn: prefix I see in some SMIL attributes?	page 139
How can I view the SMIL file for a streaming presentation?	page 141

Clips and URLs

Chapter 8 is your primary resource for learning about clip source tags and URLs. Chapter 20 contains information about servers and streaming protocols.

Clips and URLs

Question	Answer
How do I introduce a clip, such as a video, into a presentation?	page 145
Should a clip source tag have an ID?	page 146
How do I set an image clip's streaming speed?	page 146
Can I treat a SMIL or Ram file like a clip and use it in another SMIL file?	page 150
What URLs should I use in my SMIL file as I develop it?	page 151
How do I move my clips from my desktop computer to a server?	page 419
When I stream my clips, do they all need to have individual URLs?	page 152
What is Real Time Streaming Protocol (RTSP)?	page 417
How do I write an RTSP URL?	page 153
How do I write an HTTP URL?	page 153
Does RealONE Player cache files like a Web browser?	page 154
What is the CHTTP protocol?	page 154

Colors and Transparency

SMIL and RealNetworks' SMIL customizations give you many ways to add color to your presentation, as well as to modify the colors in existing clips to create transparency or partial transparency.

Colors and Transparency

Question	Answer
What color values does SMIL accept?	page 475
How do I add a background color to a region that plays a clip?	page 215
How do I make the background color appear only when the clip plays?	page 216
Can I make the region background color partially transparent?	page 216
Can I change a region background color when a clip starts?	page 217
Can I change the region background color while a clip plays?	page 347
How do I turn an entire clip partially transparent?	page 158

(Table Page 1 of 2)

Colors and Transparency (continued)

Question	Answer
How can I make a clip's opaque background transparent, or vice versa?	page 158
Can I turn a range of colors in a clip transparent?	page 159
Can I substitute a certain color for a clip's transparent background?	page 162
Can I make a clip become more (or less) transparent as it plays?	page 358
Can I create a solid block of color other than a region background color?	page 148

(Table Page 2 of 2)

Layouts

Chapter 11 explains how to lay out clips in the RealONE Player media window.

Layouts

Question	Answer
Where in the SMIL file do I define the layout?	page 202
Is a SMIL region like an HTML frame?	page 196
How do I set my presentation's overall size?	page 203
Can I make a clip play in a separate window?	page 204
How do a define the size of the region in which a clip plays?	page 208
How do I specify which clips play in which regions?	page 213
Can I play one clip in front of another?	page 214
How do I put a logo on top of my video?	page 218
How do I center a clip in a region?	page 221
How can I make my clip scale up or down to fit the region?	page 227
Can I make the same clip display in more than one region?	page 233

Basic Timing and Groups

Refer to Chapter 12 for basic information about timing presentations. Chapter 10 explains how to organize clips into groups.

Basic Timing and Groups

Question	Answer
How do I make clips play one after another?	page 175
How do I play several clips at the same time?	page 177
Can I make an entire group of clips stop when one of the clips finishes?	page 245
How do I let the viewer select which clip to play?	page 186
How do I create a clip preview?	page 242
Can I make a clip repeat?	page 247
How do I specify how long an image clip displays?	page 243
Can I make a clip play indefinitely?	page 244
Can I use timing values with groups as well as clips?	page 241
How do I make a clip freeze on screen after it stops playing?	page 251
How do I make a clip display throughout the presentation?	page 255
How do I delay when a clip starts playing?	page 240

Advanced Timing

Chapter 13 explains the advanced timing feature, which build on the basic timing features described in Chapter 12.

Advanced Timing

Question	Answer
Can I start or stop an element when any one of multiple events occur?	page 266
How do I start or stop a clip when a clip in another group starts or stops?	page 266
Can I start or stop a clip when another clip repeats?	page 268
How do I start or stop a clip when the viewer clicks an icon?	page 269
How can I coordinate all clips with a broadcast?	page 274
Can I prevent a clip from restarting?	page 274
Can I create an effect similar to a Javascript rollover?	page 269
How do I launch a clip on a keystroke?	page 271

Hyperlinks

Chapter 14 explains how to create hyperlinks in a SMIL presentation.

Hyperlinks

Question	Answer
How do I create an image map over a clip?	page 285
Can a SMIL link have an alt value?	page 293
How do I open a link with a keystroke?	page 291
How do I link my SMIL presentation to a Web page?	page 294
Can I open a link in a browser frame?	page 299
How do I open an HTML page in the RealONE Player context window?	page 297
Does RealONE Player pause when a Web page opens?	page 300
How do I open a link automatically?	page 293
How do I link my SMIL file to another streaming presentation?	page 301
Can I open a linked clip or SMIL presentation in a new window?	page 302
Can RealText include hyperlinks, too?	page 116

Special Effects

Chapter 15 and Chapter 16 explain transition effects and SMIL animations, respectively, the two features that allow you to create special effects with clips.

Special Effects

Question	Answer
How do I introduce a new clip with a transition effect?	page 317
Can I control how long a transition effect takes to complete?	page 331
How do I keep a clip visible long enough for a transition effect to occur?	page 336
Can I stop a transition effect before it completes?	page 332
How do I fade a clip to or from a solid color?	page 338
How do I fade a video into the next video?	page 339
Can I fade the volume of an audio clip up or down?	page 347
How do I select what clip and property I want to animate?	page 346
How do I make a clip grow or shrink?	page 349
How do I move a clip around the screen?	page 359

(Table Page 1 of 2)

Special Effects (continued)

Question	Answer
How do I animate colors?	page 358
How do I make an animation flow smoothly?	page 353

(Table Page 2 of 2)

Advanced Streaming

Chapter 17 and Chapter 18 cover switching and prefetching, respectively, two advanced features that allow you to stream different clips to different viewers, and maintain greater control over bandwidth use.

Advanced Streaming

Question	Answer
How do I deliver different clips to different viewers?	page 363
Do I always have to use a `<switch>` tag when I present multiple choices?	page 365
What attributes can RealONE Player evaluate when choosing a clip?	page 366
Can RealONE Player evaluate more than one attribute at a time?	page 377
How do I deliver clips in different languages to different viewers?	page 367
How do I stream clips at different bandwidths to different viewers?	page 370
Can I add captions and audio descriptions to aid viewer accessibility?	page 372
How do I deliver different clips to Windows, Macintosh, and Linux users?	page 374
Can I use a `<switch>` tag to stream different sizes of videos?	page 378
How do I stream clip data to RealONE Player before the clip plays?	page 384
How much bandwidth can I use for prefetching clip data?	page 385
How much clip data can I stream in advance of clip playback?	page 387

Web Page Embedding

See Chapter 19 for information on embedding your streaming presentation directly in a Web page.

Web Page Embedding

Question	Answer
Will an `<EMBED>` tag work with Microsoft Internet Explorer?	page 395
What is a `.rpm` file?	page 398

(Table Page 1 of 2)

Web Page Embedding (continued)

Question	Answer
How do I set my presentation's size in my Web page?	page 400
How do I add RealONE Player controls to my Web page?	page 401
How do I make all the controls work together?	page 407
Can I center my clip in an HTML table?	page 410
Can I make my presentation start as soon as the Web page loads?	page 411
How do I set shuffle play?	page 412
How do I lay out my SMIL presentation in my Web page?	page 412
Can I use Javascript or VBScript to control my embedded presentation?	page 397

(Table Page 2 of 2)

APPENDIX

C

SMIL TAG REFERENCE

Intended for advanced users, this appendix provides a reference to SMIL tags and attributes. Be sure to familiarize yourself with "Conventions Used in this Guide" on page 9, which explains the typographical conventions used in this appendix.

<smil>...</smil>

The <smil> and </smil> tags must start and end the SMIL markup. The SMIL 2.0 namespace declaration is required. You must declare the RealNetworks extension namespace if your SMIL file includes a customized attribute that uses the rn: prefix.

<smil> Tag Namespaces

Namespace	Features Defined	Reference
xmlns="http://www.w3.org/2001/SMIL20/Language"	SMIL 2 Language Profile	page 134
xmlns:rn="http://features.real.com/2001/SMIL20/Extensions"	RealNetworks extensions	page 140

Header Tags

The SMIL file header, created between <head> and </head> tags, contains tags that let you define the presentation's layout, information, transitions, and other features. For basic information about defining the SMIL file header, see "Header and Body Sections" on page 134.

<meta/>

The header region's <meta/> tags provide presentation information. A <meta/> tag can also set a base URL for source clips in the SMIL file. The content and name attributes are required for

each <meta> tag. For basic information about the <meta/> tag, see "Defining Information for the SMIL Presentation" on page 169.

<meta/> Attributes

Attribute	Value	Function	Reference
content	*text|URL*	Provides the content for the name attribute.	page 169
name	abstract	Gives the presentation abstract.	page 169
	author	Lists the presentation author's name.	page 169
	base	Sets the base URL for the source clips.	page 152
	copyright	Supplies the presentation copyright.	page 169
	title	Gives the presentation title.	page 169

Examples

```
<meta name="author" content="Jane Morales"/>
<meta name="title" content="Multimedia My Way"/>
<meta name="copyright" content="(c)2001 Jane Morales"/>
<meta name="base" content="rtsp://realserver.example.com/"/>
```

<layout>...</layout>

The <layout> and </layout> tags within the SMIL header contain other tags that define the layout of visual clips. Within the layout section, you define a root-layout area and separate regions for clips. You can also define secondary media windows.

<root-layout/>

Within the layout section, a single <root-layout/> tag sets the overall size of the main media window. Clips play in regions created within the root-layout area. They do not play in the root-layout area directly. The height and width attributes are required for the <root-layout/> tag. For basic information about the <root-layout/> tag, see "Defining the Main Media Window" on page 203.

<root-layout/> Attributes

Attribute	Value	Default	Function	Reference
backgroundColor	*color_value*	black	Sets the window background color.	page 215
height	*pixels*	(none)	Sets the main media window height.	page 203
rn:resizeBehavior	percentOnly|zoom	zoom	Controls whether regions resize.	page 206
width	*pixels*	(none)	Sets the main media window width.	page 203

Example

```
<layout>
  <root-layout backgroundColor="maroon" width="320" height="240"/>
  <region ...playback region defined.../>
  <region ...playback region defined.../>
</layout>
```

<topLayout>...</topLayout>

Following <root-layout/>, <topLayout>...</topLayout> tags can define the overall size of secondary media windows that are detached from the main media window. You assign clips to play in regions within this window. You cannot assign clips directly to a <topLayout> window. The height and width attributes are required for the <topLayout> tag. For basic information about the <topLayout> tag, see "Creating Secondary Media Windows" on page 204.

<topLayout/> Attributes

Attribute	Value	Default	Function	Reference
backgroundColor	*color_value*	black	Sets the background color.	page 215
close	onRequest|whenNotActive	onRequest	Determines when the window closes.	page 204
height	*pixels*	(none)	Sets the window height.	page 204
open	onStart|whenActive	onStart	Controls when the window opens.	page 204
rn:resizeBehavior	percentOnly|zoom	zoom	Controls whether regions resize.	page 206
width	*pixels*	(none)	Sets the window width.	page 204

Example

```
<layout>
  <root-layout.../>
  ...main media window regions defined...
  <topLayout width="180" height="120" open="whenActive" close="whenNotActive">
    ...secondary media window regions defined...
  </topLayout>
</layout>
```

<region/>

Following <root-layout/>, or between <topLayout> and </topLayout>, <region/> tags define the size, placement (relative to the window), and properties of each region used to play clips. The id

attribute is required for the <region/> tag. For basic information about the <region/> tag, see "Defining Playback Regions" on page 206.

<region/> Attributes

Attribute	Value	Default	Function	Reference
backgroundColor	inherit|transparent|*color_value*	transparent	Sets the region background color.	page 215
bottom	auto|*pixels*|*percentage*	auto	Sets the region offset from the bottom of the window.	page 208
fit	fill|hidden|meet|scroll|slice	hidden	Controls how clips fit the region.	page 227
height	auto|*pixels*|*percentage*	auto	Sets the region height.	page 208
id	*name*	(none)	Creates a target ID for assigning clips to the region.	page 207
left	auto|*pixels*|*percentage*	auto	Sets the offset from the window's left side.	page 208
rn:opacity	*percentage*	100%	Reduces background opacity.	page 216
regionName	*name*	(none)	Provides a name for certain features.	page 207
right	auto|*pixels*|*percentage*	auto	Sets the offset from the window's right side.	page 208
showBackground	always|whenActive	always	Determines when the background color appears.	page 216
soundLevel	*percentage*	100%	Cuts or boosts a clip's audio volume.	page 218
top	auto|*pixels*|*percentage*	auto	Sets the offset from the top of the window.	page 208
width	auto|*pixels*|*percentage*	auto	Defines the region width.	page 208
z-index	*number*	0	Sets the stacking order when the region overlaps another region.	page 214

Example

The following example defines both a region and a subregion:

```
<layout>
  <root-layout .../>
  <region id="video" top="5" left="5" width="240" height="180" z-index="3"
     backgroundColor="blue" showBackground="whenActive"/>
    <region id="logo" bottom="10%" right="15%" fit="fill"/>
  </region>
</layout>
```

<regPoint/>

Between the <layout> and </layout> tags, <regPoint/> tags define registration points that determine where and how clips are placed in regions. The id attribute is required for the <regPoint/> tag. For basic information about the <regPoint/> tag, see "Creating Registration Points" on page 221.

<regPoint/> Attributes

Attribute	Value	Default	Function	Reference
bottom	auto|*pixels*|*percentage*	auto	Sets the point's offset from the region's bottom border.	page 224
id	*name*	(none)	Creates an ID for assigning the point to clips.	page 221
left	auto|*pixels*|*percentage*	auto	Sets the point's offset from the region's left side.	page 224
right	auto|*pixels*|*percentage*	auto	Sets the point's offset from the region's right side.	page 224
top	auto|*pixels*|*percentage*	auto	Sets the point's offset from the region's top border.	page 224
regAlign	topLeft|topMid|topRight|midLeft|center|midRight|bottomLeft|bottomMid|bottomRight	topLeft	Specifies how clips align to the point.	page 221

Example

```
<layout>
  ...windows and regions defined...
  <regPoint id="middle" left="50%" top="50%" regAlign="center"/>
</layout>
```

<transition/>

Following the layout section, <transition/> tags define transition effects that occur when clips start or stop. The id and type attributes are required for the <transition/> tag. For basic information about the <transition/> tag, see "Defining Transition Types" on page 317.

<transition/> Attributes

Attribute	Value	Default	Function	Reference
borderColor	blend|*color_value*	black	Specifies a border color or a blended border.	page 334
borderWidth	*pixels*	0	Specifies a border width.	page 334
fadeColor	*color_value*	black	Sets a color for fades.	page 334
direction	forward|reverse	forward	Specifies the transition direction.	page 331
dur	*time_value*	1s	Defines the length of the transition effect.	page 331
endProgress	0.0-1.0	1.0	Ends the effect before it completes fully.	page 332
horzRepeat	*integer*	1	Sets a number of horizontal repetitions.	page 333
id	*name*	(none)	Creates an ID for assigning the effect.	page 317
startProgress	0.0-1.0	0.0	Starts the effect at a midway point.	page 332
subtype	*subtype_name*	(varies)	Defines an optional subtype for each type.	page 317
type	*type_name*	(none)	Specifies the main transition type.	page 317
vertRepeat	*integer*	1	Sets a number of vertical repetitions.	page 333

Example

```
<layout>
  ...windows, regions, and registration points defined...
</layout>
<transition id="sixteenBoxes" type="fourBoxWipe" subtype="cornersOut" horzRepeat="2"
  vertRepeat="2" dur="2s"/>
```

Clip Source Tags

You add clips to a presentation with one of the following source tags:

<animation/>	animation clip such as Macromedia Flash
<audio/>	audio clip such as RealAudio
<brush/>	color block used in place of a media clip
<img/>	image file in GIF, JPEG, or PNG format
<ref/>	any type of clip not covered by the other tags
<text/>	static text file

<textstream/>	streaming text clip such as RealText
<video/>	video clip such as RealVideo

Except for <brush/>, the choice of tag does not affect playback. All clip source tags can use <ref/>, for example. The src attribute is required for all clip source tags except <brush/>. For basic information about the clip souce tags, see "Creating Clip Source Tags" on page 145.

Clip Tag Attributes

Attribute	Value	Default	Function	Reference
abstract	*text*	(none)	Provides a clip abstract.	page 168
alt	*text*	(none)	Provides alternate text.	page 170
author	*text*	(none)	Lists the clip's author.	page 168
rn:background Opacity	*percentage*	100%	Adjusts background opacity.	page 158
begin	*time_value*	0s	Delays normal playback time.	page 240
bgcolor	*color_value*	(none)	Substitutes color for transparency. Use in a <param/> tag.	page 162
bitrate	*bits_per_second*	12288	Sets a static clip's streaming speed. Use in a <param/> tag.	page 146
rn:chromaKey	*color_value*	(none)	Turns selected color transparent.	page 159
rn:chromaKey Opacity	*percentage*	0%	Adds opacity to rn:chromaKey.	page 159
rn:chromaKey Tolerance	*#nnnnnn*	(none)	Widens range of rn:chromaKey.	page 160
color	*color_value*	black	Sets color in a <brush/> tag.	page 148
clipBegin	*time_value*	(none)	Specifies the clip's internal timing mark where playback begins.	page 242
clipEnd	*time_value*	(none)	Specifies the clip's internal timing mark where playback ends.	page 242
copyright	*text*	(none)	Lists the copyright for the clip.	page 168
dur	*time_value*|media| indefinite	media	Sets the total time the clip or one of its repeating cycles plays.	page 243
end	*time_value*	(none)	Sets the end time for the clip.	page 240
erase	never|whenDone	whenDone	Specifies if the clip remains when its fill period expires.	page 255
fill	auto|default|freeze| hold|remove|transition	auto| default	Determines the fill state when the clip is no longer active.	page 251

(Table Page 1 of 2)

Clip Tag Attributes (continued)

Attribute	Value	Default	Function	Reference
fillDefault	auto|freeze|hold| inherit|remove| transition	inherit	Sets a default fill for contained elements, such as animations.	page 258
id	*name*	(none)	Names clip for reference by other elements, such as animations.	page 146
longdesc	*text*	(none)	Provides a long description.	page 170
rn:mediaOpacity	*percentage*	100%	Adjusts overall transparency.	page 158
mediaRepeat	strip|preserve	preserve	Strips out native repetitions.	page 249
readIndex	*integer*	0	Determines how assistive devices read clip information.	page 171
regAlign	topLeft|topMid| topRight|midLeft| center|midRight| bottomLeft|bottomMid| bottomRight	topLeft	Specifies which part of the clip aligns to the registration point.	page 221
region	*region_ID*	(none)	Assigns the clip to a region.	page 213
regPoint	*regPont_ID*| topLeft|topMid| topRight|midLeft| center|midRight| bottomLeft|bottomMid| bottomRight	(none)	Assigns the clip to a predefined registration point, or specifies a point on the region.	page 213
reliable	true|false	false	Ensures reliable transmission.	page 148
repeatCount	*integer*|indefinite| *fractional_value*	0	Repeats the clip the specified number of times, or indefinitely.	page 247
repeatDur	*time_value*|indefinite	0s	Repeats the clip the specified amount of time.	page 247
restart	always|default|never| whenNotActive	always	Determines if the clip can replay.	page 274
src	*URL*	(none)	Provides a full or relative URL for the clip. Not used with <brush/>.	page 145
syncBehavior	canSlip|default| independent|locked	default	Sets how the clip synchronizes to its group.	page 179
title	*text*	(none)	Provides a title for the clip.	page 168
transIn	<transition/> *ID*	(none)	Assigns a starting transition effect.	page 335
transOut	<transition/> *ID*	(none)	Assigns an ending transition effect.	page 335

(Table Page 2 of 2)

Examples

```
<video id="video1" src="rtsp://realserver.example.com/media/video2.rm" region="video"
  begin="40s" clipBegin="5100ms" clipEnd="4.5min" fill="freeze"/>
<audio id="audio1" src="rtsp://realserver.example.com/media/music.rm"
  dur="10.5s" repeatCount="5"/>
<img id="image1" src="http://www.example.com/poster.jpg" region="image_region" dur="15s">
  <param name="bitrate" value="5000"/>
</img>
<brush color="blue" region="region_1" dur="5s"/>
```

<prefetch/>

Chapter 18 describes the <prefetch/> tag, which lets you download clip data before the clip plays. The <prefetch/> tag can use many SMIL timing attributes to set limits on prefetching. It also has its own attributes that control the data download. The id attribute is required.

<prefetch> Attributes

Attribute	Value	Default	Function	Reference
bandwidth	*bits_per_second*|*percentage*	100%	Sets the bandwidth used to get data.	page 385
begin	*time_value*	0s	Delays the prefetch start.	page 240
clipBegin	*time_value*	0s	Specifies the clip's internal timing mark where prefetching begins.	page 242
clipEnd	*time_value*	(none)	Specifies the clip's internal timing mark where prefetching ends.	page 242
dur	*time_value*|media|indefinite	media	Sets the duration for prefetching.	page 243
end	*time_value*	(none)	Sets the end time for prefetching.	page 240
id	*name*	(none)	Names prefetching element for reference by other SMIL elements.	page 138
mediaSize	*bytes*|*percentage*	100%	Sets how much data to fetch based on clip size. Overrides mediaTime.	page 387
mediaTime	*time_value*|*percentage*	100%	Specifies the amount of data to fetch based on the clip's duration.	page 388
restart	always|default|never|whenNotActive	always	Determines if prefetching can restart.	page 274
src	*URL*	(none)	Gives the URL to the prefetched clip.	page 145
syncBehavior	canSlip|default|independent|locked	default	Sets how <prefetch/> synchronizes to its group.	page 179

Group Tags

Chapter 10 explains the group tags that you can use to create the superstructure for your presentation's timeline.

<seq>...</seq>

The <seq> and </seq> tags play the enclosed clips in sequence. No attributes are required for a <seq> tag, which is described in "Playing Clips in Sequence" on page 175.

<seq> Attributes

Attribute	Value	Default	Function	Reference
begin	*time_value*	0s	Delays the normal group playback.	page 241
dur	*time_value*|media|indefinite	media	Sets the total time the group plays.	page 243
end	*time_value*	(none)	Sets an end time for the group.	page 241
fill	freeze|hold|remove	remove	Determines the fill state when the group is no longer active.	page 256
fillDefault	auto|freeze|hold|inherit|remove|transition	inherit	Sets a default fill for contained clips.	page 258
id	*name*	(none)	Names the group for reference by other elements.	page 138
repeatCount	*integer*|indefinite|*fractional_value*	0	Repeats the group the specified number of times, or indefinitely.	page 247
repeatDur	*time_value*|indefinite	0s	Repeats the group the specified amount of time.	page 247
restart	always|default|never|whenNotActive	always	Determines if the group can restart.	page 274
restartDefault	always|inherit|never|whenNotActive	inherit	Specifies a restart value the group passes to its elements.	page 275
syncBehavior	canSlip|default|independent|locked	default	Determines how the group synchronizes to its containing group.	page 179
syncBehavior Default	canSlip|independent|inherit|locked	inherit	Sets the default syncBehavior value for the elements the group contains.	page 182
syncTolerance	*time_value*|inherit	inherit	Creates a tolerance value for locked elements in the group.	page 185
syncTolerance Default	*time_value*	(none)	Sets a tolerance value inherited by other groups the group contains.	page 185

Example

```
<seq repeatDur="30min">
  <audio src="rtsp://realserver.example.com/one.rm"/>
  <audio src="rtsp://realserver.example.com/two.rm"/>
</seq>
```

<par>...</par>

The <par> and </par> tags make enclosed clips play at the same time. No attributes are required for a <par> tag, which is described in "Playing Clips in Parallel" on page 177.

<par> Attributes

Attribute	Value	Default	Function	Reference
abstract	*text*	(none)	Provides an abstract for the group.	page 168
author	*text*	(none)	Lists an author for the group.	page 168
begin	*time_value*	0s	Delays the normal playback time.	page 241
copyright	*text*	(none)	Lists the copyright for the group.	page 168
dur	*time_value*|media|indefinite	media	Sets the total time the group plays.	page 243
end	*time_value*	(none)	Sets an end time for the group.	page 241
endsync	all|first|id(*clip id*)|last|media	last	Determines when the group ends.	page 245
fill	freeze|hold|remove	remove	Determines the fill state when the group is no longer active.	page 256
fillDefault	auto|freeze|hold|inherit|remove|transition	inherit	Sets a default fill for contained clips.	page 258
id	*name*	(none)	Names the group for reference by other elements.	page 138
repeatCount	*integer*|indefinite|*fractional_value*	0	Repeats the group the specified number of times, or indefinitely.	page 247
repeatDur	*time_value*|indefinite	0s	Repeats the group the specified amount of time.	page 247
restart	always|default|never|whenNotActive	always	Determines if the group can restart.	page 274
restartDefault	always|inherit|never|whenNotActive	inherit	Specifies a restart value the group passes to its elements.	page 275
syncBehavior	canSlip|default|independent|locked	default	Determines how the group synchronizes to its containing group.	page 179

(Table Page 1 of 2)

<par> Attributes (continued)

Attribute	Value	Default	Function	Reference
syncBehavior Default	canSlip|independent| inherit|locked	inherit	Sets the default syncBehavior value for the elements the group contains.	page 182
syncTolerance	*time_value*| inherit	inherit	Creates a tolerance value for locked elements in the group.	page 185
syncTolerance Default	*time_value*	(none)	Sets a tolerance value inherited by other groups the group contains.	page 185
title	*text*	(none)	Lists a title for the group.	page 168

(Table Page 2 of 2)

Examples

```
<par>
  <video src="newsong.rm"/>
  <textstream src="newsong.rt"/>
</par>
```

```
<par endsync="id(text)" repeatCount="2" begin="4s">
  <video src="newsong.rm"/>
  <textstream id="text" src="newsong.rt"/>
</par>
```

<excl>...</excl>

The <excl> and </excl> tags create an exclusive group in which only one clip can play at a time. A duration is required for the <excl> tag if all elements in the group use interactive timing. For basic information on this tag, see "Creating an Exclusive Group" on page 186.

<excl> Attributes

Attribute	Value	Default	Function	Reference
begin	*time_value*	0s	Delays the normal playback time.	page 241
dur	*time_value*|media| indefinite	media	Sets the total time the group plays.	page 243
end	*time_value*	(none)	Sets an end time for the group.	page 241
endsync	all|first|id(*clip id*)| last|media	last	Determines when the group ends.	page 245
fill	freeze|hold|remove	remove	Determines the fill state when the group is no longer active.	page 256
fillDefault	auto|freeze|hold| inherit|remove| transition	inherit	Sets a default fill for contained clips.	page 258

(Table Page 1 of 2)

<excl> Attributes (continued)

Attribute	Value	Default	Function	Reference
id	*name*	(none)	Names the group for reference by other elements.	page 138
repeatCount	*integer*|indefinite|*fractional_value*	0	Repeats the group the specified number of times, or indefinitely.	page 247
repeatDur	*time_value*|indefinite	0s	Repeats the group the specified amount of time.	page 247
restart	always|default|never|whenNotActive	always	Determines if the group can restart.	page 274
restartDefault	always|inherit|never|whenNotActive	inherit	Specifies a restart value the group passes to its elements.	page 275
syncBehavior	canSlip|default|independent|locked	default	Determines how the group synchronizes to its containing group.	page 179
syncBehaviorDefault	canSlip|independent|inherit|locked	inherit	Sets the default syncBehavior value for the elements the group contains.	page 182
syncTolerance	*time_value*|inherit	inherit	Creates a tolerance value for locked elements in the group.	page 185
syncToleranceDefault	*time_value*	(none)	Sets a tolerance value inherited by other groups the group contains.	page 185

(Table Page 2 of 2)

Example

```
<excl dur="indefinite">
  <video src="video1.rm" begin="button1.activateEvent" region="video_region"/>
  <video src="video2.rm" begin="button2.activateEvent" region="video_region"/>
  <video src="video3.rm" begin="button3.activateEvent" region="video_region"/>
</excl>
```

<priorityClass>...</priorityClass>

These tags create a priority class within an exclusive group. Each priority class, which is described in "Modifying Clip Interruption Behavior" on page 188, defines the interruption behavior of clips within the exclusive group. No attributes are required for a <priorityClass> tag.

<priorityClass> Attributes

Attribute	Value	Default	Function	Reference
higher	pause|stop	pause	Sets class behavior on interruption by clips with higher priority.	page 190
id	*name*	(none)	Names the group for reference by other elements.	page 138

(Table Page 1 of 2)

<priorityClass> Attributes (continued)

Attribute	Value	Default	Function	Reference
lower	defer|never	defer	Specifies how interrupting clips with lower priority behave.	page 191
pauseDisplay	disable|hide|show	show	Sets a clip's appearance if the clip is paused.	page 192
peers	defer|never|pause|stop	stop	Controls how clips in the same class interrupt each other.	page 189

(Table Page 2 of 2)

Example

```
<excl>
  <priorityClass peers="pause">
    <video src="video1.rm" begin="button1.activateEvent" region="video_region"/>
    <video src="video2.rm" begin="button2.activateEvent" region="video_region"/>
    <video src="video3.rm" begin="button3.activateEvent" region="video_region"/>
  </priorityClass>
</excl>
```

<switch>...</switch>

The <switch> and </switch> tags, described in "Understanding Switching" on page 363, specify elements that RealONE Player chooses between based on certain criteria. No attributes are required for the <switch> tag.

<switch> Attributes

Attribute	Value	Function	Reference
id	*name*	Names the group as a link target for other SMIL files.	page 138

Test Attributes

Elements within a <switch> group must include a test attribute, such as systemBitrate or systemLanguage. You can add a test attribute to any clip source tag, as well as <a/>, <area/>,

<layout>, <region/>, <prefetch/>, <excl>, <par>, <seq>, <animate/>, <animateColor/>, <animateMotion/>, and <set/> tags.

Test Attributes for Switching

Attribute	Value	Default	Function	Reference
systemAudioDesc	on|off	off	Tests for an audio descriptions preference.	page 372
systemBitrate	*bits_per_second*	(none)	Tests for the bit rate.	page 370
systemCaptions	on|off	off	Tests for a captions preference.	page 372
systemCPU	*CPU_type*	(none)	Tests for a CPU type.	page 373
systemLanguage	*language_code*	(none)	Tests for a language preference.	page 367
systemOperatingSystem	*OS_name*	(none)	Tests for the operating system.	page 374
systemOverdubOrSubtitle	overdub|subtitle	(none)	Tests for an overdub or subtitle preference.	page 369
systemScreenDepth	1|4|8|24|32	(none)	Tests for the monitor color depth.	page 376
systemScreenSize	*pixel_height*X *pixel_width*	(none)	Tests for the monitor size.	page 376

Examples

```
<switch>
  <audio src="seattle_french.rm" systemLanguage="fr"/>
  <audio src="seattle_german.rm" systemLanguage="de"/>
  <audio src="seattle_english.rm"/>
</switch>
```

```
<switch>
  <ref src="slides1.rp" systemBitrate="80000"/>
  <ref src="slides2.rp" systemBitrate="20000"/>
</switch>
```

Hyperlink Tags

<a>...

The <a>... tags turn the enclosed clip source tag into a hyperlink. For basic information about these tags, see "Creating a Simple Link" on page 283. The href attribute is required for the <a> tag.

<a/> Attributes

Attribute	Value	Default	Function	Reference
accesskey	*key*	(none)	Sets a keystroke that opens the link.	page 291
actuate	onLoad|onRequest	onRequest	Determines whether or not the link requires user activation.	page 293
alt	*text*	(none)	Supplies alternate text for the link.	page 293
destinationLevel	*percentage*	100%	Specifies the audio level of the target.	page 306
destination Playstate	pause|play	play	Sets the play state of the target when the link opens.	page 302
external	true|false	false	Sends the link to the browser if true.	page 295
href	*URL*	(none)	Gives the link URL.	page 290
show	new|replace	replace	Sets the current or a new media window as the target.	page 302
sourceLevel	*percentage*	100%	Sets the audio level of the source.	page 306
sourcePlaystate	pause|play|stop	pause|play	Sets the play state of the source.	page 302
tabindex	*integer*	0	Sets the tabbing order for links.	page 294
target	*name*	(current window)	Identifies a window or a SMIL region.	page 299 page 303

Example

```
<a href="http://www.real.com" external="true" sourcePlaystate="pause">
  <video src="video.rm" region="videoregion"/>
</a>
```

<area/>

An <area/> tag can define a hot spot hyperlink that can be temporal as well as spatial. It fits within a binary media source tag:

```
<video ...>
  <area .../>
</video>
```

The following table lists possible <area/> tag attributes. No attributes are required for this tag, but href is typically included. For basic information about the <area/> tag, see "Using the <area/> Tag" on page 283.

<area/> Attributes

Attribute	Value	Default	Function	Reference
accesskey	*key*	(none)	Sets a keystroke that opens the link.	page 291
actuate	onLoad|onRequest	onRequest	Determines whether or not the link requires user activation.	page 293
alt	*text*	(none)	Supplies alternate text for the link.	page 293
begin	*time_value*	(none)	Sets when the link becomes active.	page 284
coords	*pixels*|*percentage*	(none)	Defines the hot spot size and location.	page 283
destinationLevel	*percentage*	100%	Specifies the audio level of the target.	page 306
destination Playstate	pause|play	play	Sets the play state of the target when the link opens.	page 302
end	*time_value*	(none)	Sets when the link deactivates.	page 284
external	true|false	false	Sends the link to the browser if true.	page 295
height	*pixels*	media height	Sets context window size in <param>.	page 297
href	*URL*	(none)	Gives the link URL.	page 290
id	*name*	(none)	Defines the element ID.	page 138
nohref	(none)	(none)	Indicates that the link has no URL.	page 291
rn:sendTo	_osdefaultbrowser| _rpbrowser| _rpcontextwin	(none)	Specifies a browser window that opens the HTML page.	page 295 page 297
shape	rect|circle|poly	rect	Sets the hotspot shape.	page 286
show	new|replace	replace	Sets the current or a new media window as the target.	page 302
sourceLevel	*percentage*	100%	Sets the audio level of the source.	page 306
sourcePlaystate	pause|play|stop	pause|play	Sets the play state of the source.	page 302
tabindex	*integer*	0	Sets the tabbing order for links.	page 294
target	*name*	current window	Identifies a window or a SMIL region.	page 299 page 303
width	*pixels*	330	Sets context window size in <param>.	page 297

Examples

```
<video src="video.rm" region="video_region">
  <area href="http://www.example.com/context.html" external="true" rn:sendTo="_rpcontextwin"
    sourcePlaystate="play">
      <rn:param name="width" value="320"/>
      <rn:param name="height" value="240"/>
  </area>
</video>
```

```
<video src="video.rm" region="video_region">
  <area href="rtsp://realserver.example.com/video2.rm" shape="circle" coords="80,60,30"
    begin="5s" end="45s" show="new" sourcePlaystate="play" destinationPlaystate="play"/>
</video>
```

Animation Tags

<animate/>

The <animate/> tag is the basic animation tag. Other animation tags are variations of <animate/>. The targetElement and attributeName attributes are generally required, as well as one of the to, by, or values attributes. For more on this tag, see "Creating Basic Animations" on page 345.

<animate/> Tag Attributes

Attribute	Value	Default	Function	Reference
accumulate	none|sum	none	Makes a repeating animation build with each iteration when set to sum.	page 356
additive	replace|sum	replace	Adds the animation value to the existing attribute value if set to sum.	page 355
attributeName	*attribute_name*	(none)	Selects the attribute to animate.	page 346
begin	*time_value*	0s	Delays normal playback time.	page 240
by	*pixels*|*percentage*|*color_value*	(none)	Animates the element by a certain amount. Do not use with to.	page 350
calcMode	discrete|linear|paced	linear	Controls the flow of an animation.	page 353
dur	*time_value*|indefinite	media	Sets the total time the animation or one of its repeating cycles plays.	page 243
end	*time_value*	(none)	Sets the end time for the animation.	page 240
fill	auto|default|freeze|hold|remove	auto|default	Determines the fill state when the animation is no longer active.	page 251

(Table Page 1 of 2)

<animate/> Tag Attributes (continued)

Attribute	Value	Default	Function	Reference
from	*pixels*|*percentage*|*color_value*	(none)	Sets a starting point for the animation. Use with to or by.	page 350
id	*name*	(none)	Names the animation for reference by other elements.	page 138
repeatCount	*integer*|indefinite|*fractional_value*	0	Repeats the animation the specified number of times, or indefinitely.	page 247
repeatDur	*time_value*|indefinite	0s	Repeats the animation the specified amount of time.	page 247
restart	always|default|never|whenNotActive	always	Determines if the animation can restart.	page 274
targetElement	*ID*	(none)	Identifies the tag that contains the animated attribute.	page 346
to	*pixels*|*percentage*|*color_value*	(none)	Sets an end point for the animation. Do not use with by.	page 350
values	*pixels*|*percentage*|*color_value*	(none)	Defines a list of values applied to the animated attribute.	page 352

(Table Page 2 of 2)

Examples

```
<animate targetElement="video_region" attributeName="width" to="380" dur="3s"/>
```

```
<animate targetElement="image_region" attributeName="width" dur="2s" by="16" accumulate="sum"
repeatCount="4" calcMode="discrete"/>
```

<animateColor/>

The <animateColor/> tag is similar to <animate/>, but it works for color animations only. The targetElement and attributeName attributes are generally required, as well as one of the to, by, or values atttributes. For more on this tag, see "Animating Colors" on page 358.

<animateColor/> Tag Attributes

Attribute	Value	Default	Function	Reference
attributeName	*attribute_name*	(none)	Selects the attribute to animate.	page 346
begin	*time_value*	0s	Delays normal playback time.	page 240
by	*pixels*|*percentage*|*color_value*	(none)	Animates the element by a certain amount. Do not use with to.	page 350
calcMode	discrete|linear|paced	linear	Controls the flow of an animation.	page 353

(Table Page 1 of 2)

<animateColor/> Tag Attributes (continued)

Attribute	Value	Default	Function	Reference
dur	*time_value*|indefinite	media	Sets the total time the animation or one of its repeating cycles plays.	page 243
end	*time_value*	(none)	Sets the end time for the animation.	page 240
fill	auto|default|freeze|hold|remove	auto|default	Determines the fill state when the animation is no longer active.	page 251
from	*pixels*|*percentage*|*color_value*	(none)	Sets a starting point for the animation. Use with to or by.	page 350
id	*name*	(none)	Names the animation for reference by other elements.	page 138
restart	always|default|never|whenNotActive	always	Determines if the animation can restart.	page 274
targetElement	*ID*	(none)	Identifies the tag that contains the animated attribute.	page 346
to	*pixels*|*percentage*|*color_value*	(none)	Sets an end point for the animation. Do not use with by.	page 350
values	*pixels*|*percentage*|*color_value*	(none)	Defines a list of values applied to the animated attribute.	page 352

(Table Page 2 of 2)

Example

```
<animateColor targetElement="image_region" attributeName="backgroundColor"
values="red;blue;yellow" calcMode=”discrete” begin="1s" dur="12s" fill="freeze"/>
```

<animateMotion/>

The <animateMotion/> tag can move an element both horizontally and vertically. The targetElement attribute is generally required, as well as one of the to, by, or values atttributes. For more on this tag, see “Creating Horizontal and Vertical Motion” on page 359.

<animateMotion/> Tag Attributes

Attribute	Value	Default	Function	Reference
accumulate	none|sum	none	Makes a repeating animation build with each iteration when set to sum.	page 356
additive	replace|sum	replace	Adds the animation value to the existing attribute value when set to sum.	page 355
begin	*time_value*	0s	Delays normal playback time.	page 240

(Table Page 1 of 2)

<animateMotion/> Tag Attributes (continued)

Attribute	Value	Default	Function	Reference
by	*pixels*|*percentage*|*color_value*	(none)	Animates the element by a certain amount. Do not use with to.	page 350
calcMode	discrete|linear|paced	paced	Controls the flow of an animation.	page 353
dur	*time_value*|indefinite	media	Sets the total time the animation or one of its repeating cycles plays.	page 243
end	*time_value*	(none)	Sets the end time for the animation.	page 240
fill	auto|default|freeze|hold|remove	auto|default	Determines the fill state when the animation is no longer active.	page 251
from	*pixels*|*percentage*|*color_value*	(none)	Sets a starting point for the animation. Use with to or by.	page 350
id	*name*	(none)	Names the animation for reference by other elements.	page 138
repeatCount	*integer*|indefinite|*fractional_value*	0	Repeats the animation the specified number of times, or indefinitely.	page 247
repeatDur	*time_value*|indefinite	0s	Repeats the animation the specified amount of time.	page 247
restart	always|default|never|whenNotActive	always	Determines if the animation can restart.	page 274
targetElement	*ID*	(none)	Identifies the tag that contains the animated attribute.	page 346
to	*pixels*|*percentage*|*color_value*	(none)	Sets an end point for the animation. Do not use with by.	page 350
values	*pixels*|*percentage*|*color_value*	(none)	Defines a list of values applied to the animated attribute.	page 352

(Table Page 2 of 2)

Example

```
<animateMotion targetElement="image_region" values="180,180;60,340;125,95"
calcMode="discrete" begin="7s" dur="5s" fill="freeze"/>
```

<set/>

The <set/> tag sets an attribute to a specified value. The targetElement, attributeName, and to attributes are required. For more on this tag, see "Setting an Attribute Value" on page 360.

<set/> Tag Attributes

Attribute	Value	Default	Function	Reference
attributeName	*attribute_name*	(none)	Selects the attribute to animate.	page 346
begin	*time_value*	0s	Delays normal playback time.	page 240
dur	*time_value*|indefinite	media	Sets the total time the animation or one of its repeating cycles plays.	page 243
end	*time_value*	(none)	Sets the end time for the animation.	page 240
fill	auto|default|freeze|hold|remove	auto|default	Determines the fill state when the animation is no longer active.	page 251
id	*name*	(none)	Names the animation for reference by other elements.	page 138
restart	always|default|never|whenNotActive	always	Determines if the animation can restart.	page 274
targetElement	*ID*	(none)	Identifies the tag that contains the animated attribute.	page 346
to	*pixels*|*percentage*|*color_value*	(none)	Sets the attribute value.	page 350

Example

```
<set targetElement="video_region" attributeName="backgroundColor" to="blue" dur="30s"/>
```

APPENDIX

D

SMIL COLOR VALUES

SMIL supports the color values defined in the Cascading Style Sheets 2 (CSS2) specification. For any SMIL color attribute, you can use a predefined color name, a hexadecimal value, or an RGB value, letting you specify millions of colors.

> **Note:** Markup other than SMIL may not support all the color designations described in this appendix. RealText, for example, supports color names and hexadecimal values, but not RGB values.

> **For More Information:** The CSS2 color specification is located at **http://www.w3.org/TR/REC-CSS2/syndata.html#value-def-color**.

Specifying RGB Color Values

Any SMIL color attribute accepts a red/green/blue (RGB) value, as shown in the following example:

```
backgroundColor="rgb(128,56,10)"
```

> **Tip:** Spaces between the color values is OK, so `rgb(128, 56, 10)` works, too.

Using Standard RGB Color Values

In the RGB color scheme, there are 256 possible values for each of the red, green, and blue components of a color pixel on a computer screen. In RGB notation, each color value ranges from 0 (no color) to 255 (full color). A full color value combines a red, a green, and a blue value. Here are a few examples of RGB color values:

- `rgb(0,0,0)` is black

- rgb(255,0,0) is bright red
- rgb(255,255,0) is bright yellow
- rgb(0,0,255) is bright blue
- rgb(255,255,255) is white

Specifying RGB Percentages

SMIL also supports percentage values for RGB coordinates, in which 0% corresponds to the value 0, and 100% corresponds to the value 255. Here is an example that is equivalent to rgb(25,191,103):

```
backgroundColor="rgb(10%,75%,40.5%)"
```

> **Tip:** Decimal values are acceptable for percentages. In all cases, RealONE Player converts the percentage values to their closest RGB equivalents.

Defining Hexadecimal Color Values

You can specify any RGB color with a hexadecimal (base 16) value and a leading pound sign (#), as shown in the following example:

```
backgroundColor="#34F9A8"
```

Hexadecimal numbering uses the digits 0 through 9, along with the "digits" A through F. Decimal 5 and hexadecimal 5 are the same value, for example, but decimal 10 corresponds to hexadecimal A, decimal 15 corresponds to hexadecimal F, and decimal 16 corresponds to hexadecimal 10.

Using Six-Digit Hexadecimal Values

Hexadecimal color values are typically six digits, in which the first pair of digits defines an RGB red value, the second pair specifies a green value, and the last pair specifies a blue value. Each hexadecimal pair can specify 256 colors (16 x 16), thereby replicating the RGB single-color values of 0 to 255. Each hexadecimal red, green, or blue color value ranges from 00 (no color) to FF (full color). Here are some examples:

- #000000 is black
- #FF0000 is bright red
- #FFFF00 is bright yellow

- `#0000FF` is bright blue
- `#FFFFFF` is white

> **Note:** Letter case does not matter for hexadecimal digits. Hence, `#ACBD5F` is equivalent to `#acbd5f`.

Defining a Three-Digit Hexadecimal Value

In place of any six-digit hexadecimal value, you can use a three-digit value, in which each digit specifies a red, green, and blue RGB value, respectively:

```
backgroundColor="#3F8"
```

The three-digit value is converted to a six-digit value by duplicating each digit. The preceding three-digit value is therefore equivalent to the following value:

```
backgroundColor="#33FF88"
```

> **Tip:** Using the three-digit notation, you can quickly specify white (`#FFF`) or black (`#000`).

Using Color Names

The simplest, but most limited, way to specify a color is to use a predefined color name, as shown in the following example:

```
backgroundColor="blue"
```

SMIL 2.0, CSS2, and HTML 4.0 all support the same 16 predefined color names, which are listed in the following table. Each color name's hexadecimal and RGB color value is included as reference, but you specify only the name when defining the color.

white
#FFFFFF
rgb(255,255,255)
	silver
	#C0C0C0
	rgb(192,192,192)
		gray
		#808080
		rgb(128,128,128)
			black
			#000000
			rgb(0,0,0)

yellow
#FFFF00
rgb(255,255,0)
	fuchsia
	#FF00FF
	rgb(255,0,255)
		red
		#FF0000
		rgb(255,0,0)
			maroon
			#800000
			rgb(128,0,0)

lime #00FF00 rgb(0,255,0)	**olive** #808000 rgb(128,128,0)	**green** #008000 rgb(0,128,0)	**purple** #800080 rgb(128,0,128)
aqua #00FFFF rgb(0,255,255)	**teal** #008080 rgb(0,128,128)	**blue** #0000FF rgb(0,0,255)	**navy** #000080 rgb(0,0,128)

Tips for Defining Color Values

- Both the RGB and hexadecimal color schemes let you define the same colors. Use whichever method you prefer.
- Illustration programs typically define colors uisng the RGB scheme, while the hexadecimal scheme is common to HTML markup programs. Many newer programs support both schemes, though, and let you convert easily between them. Web resources are also available to convert an RGB value to hexadecimal, and vice versa.
- You can mix color names, RGB values, and hexadecimal values within a SMIL file, using RGB for some attributes and hexadecimal values for others, for example.
- Most color monitors can display all the colors that you can define through SMIL. If a monitor cannot display the full range of colors, it displays the nearest approximations.
- Keep in mind that some viewers may be color blind (especially between greens and reds), or may not be able to discriminate between subtle color differences. It's a good idea always to use highly contrasting colors, such as bright, light text on a dark background.

APPENDIX

E

REALTEXT TAG REFERENCE

Use this appendix for reference when writing RealText files. For complete information on RealText, see Chapter 6. In the tables throughout this appendix, an asterisk (*) denotes a required attribute. Ellipses (...) indicate where nonessential information has been left out of examples.

Window Tag Attributes

The <window> tag that starts each RealText clip can use the attributes specified in the following table to set the overall clip parameters.

RealText <window> Tag Attributes

Attribute	Possible Values	Default	Function	Reference
bgcolor	*name*|#*RRGGBB*| transparent	black (tickertape) white (all others)	Sets the window color.	page 96
crawlrate	pixels per second	20 (tickertape) 20 (marquee) 0 (all others)	Sets the horizontal text speed.	page 100
duration	*hh:mm:ss.xy*	60 seconds	Specifies presentation length.	page 97
extraspaces	use|ignore	use	Recognizes or ignores extra spaces in text.	page 102
height	pixels	30 (tickertape) 30 (marquee) 180 (all others)	Sets the window pixel height.	page 96
link	*name*|#*RRGGBB*	blue	Specifies the hyperlink color.	page 100
loop	true|false	true (tickertape) true (marquee) false (all others)	Turns text looping on or off.	page 101
scrollrate	pixels per second	10 (scrollingnews) 0 (all others)	Sets the vertical text speed.	page 100

(Table Page 1 of 2)

RealText <window> Tag Attributes

Attribute	Possible Values	Default	Function	Reference
type	generic|tickertape|marquee|scrollingnews|teleprompter	generic	Sets the window type.	page 94
underline_hyperlinks	true|false	true	Determines whether hyperlinks are underlined.	page 100
version	1.0|1.2|1.4|1.5	1.0	Specifies RealText version. Required for some character sets.	page 99
width	pixels	500 (tickertape) 500 (marquee) 320 (all others)	Sets the window pixel width.	page 96
wordwrap	true|false	true	Turns word wrap on or off.	page 101

(Table Page 2 of 2)

Example

```
<window type="scrollingnews" width="218" height="420" bgcolor="green" version="1.5"
duration="180.5" underline_hyperlinks="false" link="red">
...all clip text...
</window>
```

Time and Position Tags

The tags in the following table let you time and position the text in a RealText clip.

RealText Time and Position Tags

Tag	Attributes	Default	Function	Reference
<clear/>	(none)	(none)	Clears all text from the window.	page 104
<pos/>	x="*pixels*"|y="*pixels*"	(none)	Positions text.	page 105
<required>...</required>	(none)	(none)	Ensures that text is delivered.	page 106
<time/>	begin="*hh:mm:ss.xy*"|end="*hh:mm:ss.xy*"	(none)	Sets time when text appears or disappears.	page 102
<tl>...</tl>	color="*name*|*#RRGGBB*"	green	Place text at bottom of ticker tape.	page 105
<tu>...</tu>	color="*name*|*#RRGGBB*"	white	Place text at bottom of ticker tape.	page 105

Examples

```
<time begin="10"/>Display at 10 seconds after clip starts.
<time begin="15"/><clear/>Clear previous text and display at 15 seconds after clip starts.
```

```
<tu color="yellow">DJIA</tu>
<tl color="blue">7168.35 +36.52 </tl>
```

Layout and Appearance Tags

The following RealText tags affect the layout and appearance of text.

RealText Layout and Apperance Tags

Tag	Function	Reference
...	**Bolds** the enclosed text.	page 115
 	Creates a line break and displays text one line down.	page 114
...	Centers the enclosed text.	page 114
	Acts like two tags, but does not create a horizontal rule. Provided for HTML compatibility.	page 115
...	*Italicizes* the enclosed text.	page 115
...	Acts like a tag. Provided for HTML compatibility.	page 115
...	Indents text, but does not number it. Provided for HTML compatibility.	page 115
...	Creates a text paragraph.	page 114
<pre>...</pre>	Displays text in a monospace font and preserves extra spaces. Works the same as in HTML.	page 114
<s>...</s>	~~Strikes through~~ the enclosed text.	page 115
<u>...</u>	Underlines the enclosed text.	page 115
...	Indents text, but does not add bullets to it. Provided for HTML compatibility.	page 115

Examples

```
<center>This is centered text.</center>
```

```
This is <b>bolded</b> text.
```

```
This is <u>underlined</u> text.
```

Font Tag Attributes

The <font> tag lets you select fonts and character sets.

RealText <font> Tag Attributes

Attribute	Possible Values	Default	Function	Reference
bgcolor	*name*|#*RRGGBB*	transparent	Sets the text background color.	page 112
charset	us-ascii|iso-8859-1| mac-roman|x-sjis| gb2312|big5|iso-2022-kr	iso-8859-1	Specifies character set used to display text.	page 106
color	*name*|#*RRGGBB*	(none)	Controls font color, except for TickerTape window.	page 112
face	(see font tables)	Times New Roman	Sets the text face.	page 109
size	-2|-1|+0|+1|+2|+3|+4 or 1|2|3|4|5|6|7	+0	Sets the font size.	page 111

Examples

```
<font color="red" bgcolor="green">This is red text against a green background.</font>
<font size="+1">This text is one size larger than the preceding text.</font>
<font face="Verdana">This text is in the Verdana font.</font>
<font charset="iso-2022-kr" face="BatangChe">...Korean text...</font>
```

Hyperlinking Commands

You can use <a> and tags to create a link out of enclosed text. The link can open a URL in RealONE Player or the viewer's browser, open an e-mail message, or issue a RealONE Player command.

RealText <a> Tag Attributes

Attribute	Possible Values	Function	Reference
href="*command*" target="_player"	command:seek(*time*) command:pause() command:play()	Creates hyperlink that issues a command.	page 117
href="command:openwindow()"	*name*|*URL*| zoomlevel	Opens new, named media windows for URL	page 306
href="mailto:*address*"	email address	Opens e-mail message.	page 116
href="*URL*"	target="_player"	Creates hyperlink to URL.	page 116

Examples

send e-mail

Play Next Clip

Visit RealGuide

Send Me an Instant Message

Seek

Play

Comedy Hour

APPENDIX

F

FILE TYPE REFERENCE

The following tables provide a quick reference to file types commonly used in RealSystem streaming. This is not a definitive list of all file types, though. Plug-in technology allows RealSystem to stream virtually any file type.

RealSystem Standard Streaming Clip Types

Extension	File Type	Reference
.rm or .ra	RealAudio	"Understanding RealAudio" on page 43
.rm	RealVideo	"Understanding RealVideo" on page 57
.rp	RealPix streaming image markup	"Images" on page 28
.rt	RealText streaming text	"Text" on page 29
.swf	Flash Player file	"Flash Animation" on page 75

RealSystem Information Files

Extension	File Type	Reference
.ram	Ram file to launch RealONE Player	"Launching RealONE Player with a Ram File" on page 420
.rpm	Ram file for embedded presentations	
.smil, .smi	SMIL file for layout and timing	"Creating a SMIL File" on page 133

Image Files Types Playable Directly in RealONE Player and RealPix

Extension	File Type	Reference
.gif	GIF87, GIF89, or animated GIF image	"Images" on page 28
.jpg	JPEG (nonprogressive) image	
.png	PNG image	

APPENDIX

G

LANGUAGE CODES

As "Switching Between Language Choices" on page 367 explains, SMIL can list different language choices that RealONE Player chooses from based on its language preference. The following table lists the codes you can use in a SMIL file to indicate clips created for specific languages.

Code	Language
af	Afrikaans
sq	Albanian
ar-iq	Arabic (Iraq)
ar-dz	Arabic (Algeria)
ar-bh	Arabic (Bahrain)
ar-eg	Arabic (Egypt)
ar-jo	Arabic (Jordan)
ar-kw	Arabic (Kuwait)
ar-lb	Arabic (Lebanon)
ar-ly	Arabic (Libya)
ar-ma	Arabic (Morocco)
ar-om	Arabic (Oman)
ar-qa	Arabic (Qatar)
ar-sa	Arabic (Saudi Arabia)
ar-sy	Arabic (Syria)
ar-tn	Arabic (Tunisia)
ar-ae	Arabic (U.A.E.)
ar-ye	Arabic (Yemen)
eu	Basque
bg	Bulgarian

Code	Language
ca	Catalan
zh-hk	Chinese (Hong Kong)
zh-cn	Chinese (People's Republic)
zh-sg	Chinese (Singapore)
zh-tw	Chinese (Taiwan)
hr	Croatian
cs	Czech
da	Danish
nl	Dutch (Standard)
nl-be	Dutch (Belgian)
en	English
en-au	English (Australian)
en-bz	English (Belize)
en-gb	English (British)
en-ca	English (Canadian)
en	English (Caribbean)
en-ie	English (Ireland)
en-jm	English (Jamaica)
en-nz	English (New Zealand)
en-za	English (South Africa)

Code	Language
en-tt	English (Trinidad)
en-us	English (United States)
et	Estonian
fo	Faeroese
fi	Finnish
fr-be	French (Belgian)
fr-ca	French (Canadian)
fr-lu	French (Luxembourg)
fr	French (Standard)
fr-ch	French (Swiss)
de-at	German (Austrian)
de-li	German (Liechtenstein)
de-lu	German (Luxembourg)
de	German (Standard)
de-ch	German (Swiss)
el	Greek
he	Hebrew
hu	Hungarian
is	Icelandic
in	Indonesian
it	Italian (Standard)
it-ch	Italian (Swiss)
ja	Japanese
ko	Korean
ko	Korean (Johab)
lv	Latvian
lt	Lithuanian
no	Norwegian
pl	Polish
pt-br	Portuguese (Brazilian)
pt	Portuguese (Standard)
ro	Romanian

Code	Language
sr	Serbian
sk	Slovak
sl	Slovenian
es-ar	Spanish (Argentina)
es-bo	Spanish (Bolivia)
es-cl	Spanish (Chile)
es-co	Spanish (Colombia)
es-cr	Spanish (Costa Rica)
es-do	Spanish (Dominican Republic)
es-ec	Spanish (Ecuador)
es-sv	Spanish (El Salvador)
es-gt	Spanish (Guatemala)
es-hn	Spanish (Honduras)
es-mx	Spanish (Mexican)
es-ni	Spanish (Nicaragua)
es-pa	Spanish (Panama)
es-py	Spanish (Paraguay)
es-pe	Spanish (Peru)
es-pr	Spanish (Puerto Rico)
es	Spanish (Spain)
es-uy	Spanish (Uruguay)
es-ve	Spanish (Venezuela)
sv	Swedish
sv-fi	Swedish (Finland)
th	Thai
tr	Turkish
uk	Ukrainian
vi	Vietnamese

GLOSSARY

A

Advertising Application
A RealServer option that inserts ads into SMIL-based presentations. SMIL files use `<RealAdInsert/>` tags to indicate ad placement.

artifact
A visual imperfection in an encoded video clip. Too many artifacts can make the video look blocky.

B

bandwidth
The upper limit on the amount of data, typically expressed as kilobits per second (Kbps), that can pass through a network connection.

banner ad
An ad that appears alongside a requested clip or presentation. A banner ad can also rotate, making new ad images appear at regular intervals.

binary tag
A SMIL tag that comprises opening and closing tags, such as `<ref...>` and `</ref>`. Many unary tags can become binary tags when necessary to enclose other tags.

bit
The smallest unit of measure of data in a computer. A bit has a binary value, either `0` or `1`.

bit rate
A measure of bandwidth, expressed as the number of bits transmitted per second. A 28.8 Kbps modem, for example, can transmit or receive around 29,000 bits per second.

blank time
A period during a presentation in which RealONE Player is not paused, but no activity occurs onscreen. You typically insert blank time with the SMIL `begin` attribute.

broadcast
To deliver a presentation, whether live or prerecorded, in which all viewers join the presentation in progress. Contrast to *on-demand*.

buffering
The receiving and storing of data before it is played back. A clip's initial buffering is called *preroll*. After this preroll, excessive buffering may stall the presentation.

byte
A common measurement of data. One byte consists of 8 bits.

C

cable modems
Devices that allow rapid transmission and reception of data over television cable. They are digital devices, unlike dial-up modems, which transmit analog data.

camel case
A capitalization convention in which words in a phrase are joined, and each word after the first begins with a capital letter. SMIL 2.0 attributes and values generally use camel case, as in `soundLevel` or `whenNotActive`.

CBR
Constant Bit Rate. A type of RealVideo encoding in which all parts of the video play back at the same bit rate. Contrast to *VBR*.

CHTTP
A version of HTTP supported by RealONE Player. Files designated with `chttp://` are downloaded through HTTP and stored in RealONE Player's cache.

client
A software application that receives data from a server. A Web browser is a client of a Web server. RealONE Player is a client of RealServer.

clip
A media file within a presentation. Clips typically have an internal timeline, as with RealAudio and RealVideo.

codec
Coder/decoder. Codecs convert data between uncompressed and compressed formats, reducing the bandwidth a clip consumes.

D

download
To send a file over a network with a nonstreaming protocol such as HTTP. Contrast to *stream*.

DSL
Digital Subscriber Line. A technology for transmitting digital data over a regular telephone line at speeds much faster than dial-up modems.

duress stream
A low-bandwidth SureStream audio or video stream that RealServer uses if a connection's available bandwidth drops greatly.

E

encoding
Converting a file into a compressed, streaming format. For example, you can encode WAV files as RealAudio clips.

F

Flash
A software application and an animation format created by Macromedia. RealONE Player can play Flash animations and stream them in parallel with other clips, such as RealAudio clips.

Flash Player file
A compressed Flash file format (file extension `.swf`) suitable for streaming. To stream Flash, you export the Flash Player file and tune it so that it plays well in RealONE Player.

fps
Frames Per Second. The number of video frames that displays each second in a streaming video clip.

frequency response
A measure of audio clip quality. The higher a clip's frequency response, the more frequencies it can faithfully reproduce.

H

HTTP
Hypertext Transport Protocol. The protocol used by Web servers to communicate with Web browsers. In contrast, RealServer streams clips to RealONE Player with RTSP. See also *CHTTP*.

I

inline switching
Switching between alternative clips without using a SMIL `<switch>` tag.

interstitial ad
An advertisement that interrupts playback of a requested clip. It is like a commercial break on television.

ISDN
Integrated Services Digital Network. Technology that makes digital data connections at 64 or 112 Kbps possible over telephone lines.

ISP
Internet Service Provider. A company that provides access to the Internet. Many ISPs have RealServer available to stream media clips.

K

kilobit (Kb)
A common unit of data measurement equal to 1024 bits. A kilobit is usually referred to in the context of bit rate per unit of time, such as kilobits per second (Kbps).

kilobyte (KB)
A common unit of data measurement equal to 1024 bytes or 8 kilobits.

L

LAN
Local Area Network. A computer network confined to a local area, such as a single building. LANs vary in speed, with bandwidth shared among all networked devices.

lossy
A compression scheme that lowers clip size by discarding nonessential data from the source file. Both RealAudio and RealVideo are lossy.

M

mouseover
The action of moving a computer screen pointer over an interactive area. An animated button may change appearance on a mouseover, for example.

N

namespace
An XML declaration that identifies the features used in a SMIL presentation. For SMIL 2.0 and higher, the `<smil>` tag must declare a namespace.

O

on-demand
A type of streaming in which a clip plays from start to finish when a user clicks a link. Most clips are streamed this way. Contrast to *broadcast*.

P

PNA
A proprietary protocol RealServer uses for backward compatibility with RealSystem 3 through 5. URLs using PNA start with `pnm://`.

port
A connection to a server, designated by a number such as 8080. RealServer uses different ports for the RTSP, HTTP, and PNA protocols.

prefetch
To stream clip data to RealONE Player before the clip plays back. A clip's preroll can be prefetched minutes before the clip plays, for example, masking the preroll from the viewer.

preroll
Buffering that occurs just before a clip plays back. Preroll should be no more than 15 seconds.

presentation
A group of clips coordinated through SMIL and streamed from RealServer to RealPlayer G2.

R

RDT
The proprietary data package protocol RealServer uses (along with RTSP) when communicating with RealONE Player. Contrast to *RTP*.

RealAudio
A RealSystem clip type for streaming audio over a network. RealAudio clips use the `.rm` extension.

RealONE Player
The successor to RealPlayer 8, the RealONE Player combines streaming and digital download technologies. It supports the SMIL 2.0 standard.

RealPix
A RealSystem clip type (file extension `.rp`) for streaming still images over a network. RealPix uses a markup language for creating special effects such as fades and zooms.

RealPlayer G2
The RealNetworks client software that introduced plug-ins and the ability to update itself. It, along with the later RealPlayer 7 and RealPlayer 8, supports the SMIL 1.0 standard.

RealProducer
The primary RealNetworks tool for encoding RealAudio and RealVideo clips.

RealServer
RealNetworks server software used to stream multimedia presentations to RealPlayer G2.

RealServer administrator
The person in charge of setting up and running RealServer.

RealSlideshow
A RealNetworks tool for creating streaming slideshows based on the RealPix markup.

RealSystem
The RealNetworks system for streaming media such as RealAudio and RealVideo clips over a network. It consists of RealServer, RealPlayer G2, and various production tools.

RealText
A RealSystem clip type (file extension .rt) for streaming text over a network. It uses a markup language for formatting text.

real-time
Delivered as it occurs. For example, a live event is streamed across a network in a real-time broadcast.

RealVideo
A RealSystem clip type for streaming video over a network. RealVideo clips use the extension `.rm`.

rotating banner ads
Banner advertisements that change at a specified interval, such as a new ad banner every 30 seconds.

RTP
Real-Time Protocol. The open, standards-based data package protocol RealServer uses (along with RTSP) to communicate with RTP-based clients. Contrast to *RDT*.

RTSP
Real-Time Streaming Protocol. An open, standards-based control protocol that RealServer uses to stream clips to RealONE Player or any RTP-based client. Contrast to *HTTP*.

S

server
1. A software application, such as a Web server or RealServer, that sends requested data over a network.

2. A computer that runs server software.

Shockwave Flash
See *Flash Player file*.

SMIL
Synchronized Multimedia Integration Language. A markup language for specifying how and when each clip plays within a presentation. SMIL files use the extension `.smil`.

stream
1. To send a media clip over a network so that it begins playing back as quickly as possible.
2. A flow of a single type of data, measured in kilobits per second (Kbps). A RealVideo clip's soundtrack is one stream, for example.

SureStream
A RealNetworks technology that enables a RealAudio or RealVideo clip to stream at multiple bit rates. Available only in RealSystem.

U

unary tag
A SMIL tag that includes a closing slash, as in `<ref…/>`. Many unary tags can become binary tags when necessary to enclose other tags.

URL
Uniform Resource Locator. A location description that enables a Web browser or RealONE Player to receive a clip stored on a Web server or RealServer.

V

VBR
Variable Bit Rate. A type of RealVideo encoding that enables RealONE Player to play different parts of the video at different bit rates, even though the video is being streamed at a constant rate. Contrast to *CBR*.

X

XML
Extensible Markup Language. The parent language for SMIL. XML allows one to develop flexible, standardized languages for any purpose.

INDEX

A

<a> tag, 283
abstract attribute, 168
access keys
  for clip timing, 271
  for hyperlinks, 291
accessibility features
  alternate description, 170
  audio descriptions, 372
  long description, 170
  read order indexing, 171
  system captions, 372
accesskey attribute
  clip timing, 271
  hyperlinks, 291
accumulate attribute, 356
activateEvent value, 270
ActiveX control, 397
actuate attribute, 293
additive attribute, 355
advertising on RealGuide, 431
AIFF conversion to RealAudio, 54
alt attribute
  clips, 170
  hyperlinks, 293
<animate/> tag, 345
<animateColor/> tag
  compared to <animate/>, 342
  using, 358
animated GIF timing, 249
<animateMotion/> tag
  compared to <animate/>, 342
  using, 359
animation
  see Flash
  see SMIL animation
<animation/> tag, 145
<area/> tag, 283
attributeName attribute, 346
AU conversion to RealAudio, 54
audio
  cables, 51
  capture cards, 26
  DC offset, 53
  digitizing, 52
  dynamics compression, 53
  editing programs, 26
  equipment quality, 51
  for Flash, 79
  frequency equalization, 54
  gain compression, 52
  input levels, 51
  normalization, 53
  optimizing, 52
  production tools, 26
  recording tips, 50
  sampling width, 52
  signal-to-noise ratio, 51
  source media, 51
  streaming steps, 49
  volume control in regions
    example, 231
    setting, 218
  see also RealAudio
audio descriptions, 372
<audio/> tag, 145
author attribute, 168
Authoring Kit, 10
autoupdate of RealONE Player, 29
AVI
  compressed, 68
  converting to RealVideo, 70

B

backgroundColor attribute
animating, 347, 349
clip source tag, 220
regions, 215
see also regions
backgroundOpacity attribute
animating, 349
using, 158
backward compatibility
RealONE Player clip support, 29
through Ram file, 421
through Ramgen, 427
bandwidth
clip characteristics
Flash, 76
images, 146
RealAudio, 44
RealPix, 34
RealText, 93
RealVideo, 57
SMIL, 34
leaving for other processes, 33
multiclip presentations, 33
negotiation, 35
network connection speeds, 32
overview, 31
preroll, 32
rebuffering, 32
repeating clips, 250
SMIL switching, 370
SureStream clips, 35, 371
switching, 371
timeline considerations, 38
bandwidth attribute, 385
begin attribute
in clips, 240
in groups, 241
in hyperlinks, 284
beginEvent value, 267
Betacam video, 66
bit rate, *see* bandwidth
bitrate parameter, 147
borderColor attribute, 334
borderWidth attribute, 334
bottom attribute
<region/> tag, 208
<regPoint/> tag, 224
animating, 347, 349
clip source tag, 220
broadcasting
audio volumes, 52
RealGuide listings, 431
RealText, 94
stream synchronization, 274
<brush/> tag, 148
by attribute, 351

C

cable modem bandwidth targets, 32
cable shielding, 51
caching
authoring example, 155
cache directory, 156
cache size, 156
CHTTP protocol, 154
control commands, 155
expiration rules, 156
overriding, 156
requirements, 154
calcMode attribute, 353
camel case, 136
captions
filler clip, 380
RealONE Player preference for, 372
resizing for captions off, 381
capture cards, 26
centering clips in regions
example, 230
through clip source tag, 222
with registration point, 225
character sets for RealText, 106
chromaKey attribute, 159
chromaKeyOpacity attribute, 159
chromaKeyTolerance attribute, 160
CHTTP, 154
clicking a clip to start or stop another clip, 269
clip information, 168
clip position and fit, 198
clip source tags

clip type indicators, 145
IDs, 146
linking
absolute file syntax, 151
base target, 152
local files, 151
RealServer, 153
relative file syntax, 151
Web server, 153
Ram file as a clip, 149
SMIL file as a clip, 150
layouts, 150
timing, 150
clipBegin attribute, 242
clipEnd attribute, 242
clock wipes, 323
close attribute, 204
codecs
see RealVideo
see RealAudio
color attribute
animating, 349
in <brush/> tag, 148
color depth switching, 376
colors
animating, 358
RealText, 112
region backgrounds, 215
compression
audio dynamics, 53
overview, 27
RealAudio, 43
RealVideo, 57
context window
background color, 298
content caching, 299
defining, 297
frames, 299
linking from browser window, 299
linking to browser window, 297
overview, 280
recommended use, 298
scroll bars, 299
sizing, 297
defaults, 298
persistence, 298
recommendations, 298
relationship to media window, 298
coords attribute, 285
copyright attribute, 168
copyright protection
with RealServer, 30
with Web server, 428
CPU
guidelines, 431
switching, 373
CSS2 standard
color values, 475
SMIL positioning similarities, 209

D

de-interlace filter, 72
destinationLevel attribute, 306
destinationPlaystate attribute, 301
digital rights management, 4
digital video formats, 66
direction attribute, 331
documentation library, 10
doubling clip sizes, 422
download icon for RealONE Player, 432
downloading
RealONE Player plug-ins, 29
versus streaming, 417
DSL bandwidth targets, 32
dur attribute, 243
in repeating clip, 248
in transition effects, 331

E

edge wipes, 318
<EMBED>tag, 398
embedded playback
aspect ratio, 410
automatic playback, 411
background color, 409
backwards compatibility, 402
basics, 398
centering clip, 410
consoles, 407
image window, 402
laying out presentations, 412

local file links, 398
logo suppression, 410
looping playback
indefinitely, 411
specific number of times, 412
nonembedded links, 401
parameters
AUTOSTART, 411
BACKGROUNDCOLOR, 409
CENTER, 410
CONSOLE, 407
CONTROLS, 401
HEIGHT, 400
LOOP, 411
MAINTAINASPECT, 410
NOJAVA, 400
NOLOGO, 410
NUMLOOP, 412
REGION, 413
SHUFFLE, 412
SRC, 398
WIDTH, 400
RealONE Player controls
adding to page, 401
linking multiple controls, 407
shuffling playback, 412
size parameters
percentages, 400
pixels, 400
source parameter, 398
supported browsers, 395
URL handling, 400
using in Web page, 395
encoding tools, *see* RealProducer
end attribute
in clips, 241
in groups, 241
in hyperlinks, 284
in repeating clip, 248
endEvent value, 267
endsync attribute, 245
erase attribute, 255
error correction, 73
example files, 5
<excl> tag, 186
exclusive groups
begin and end times, 241
clip interruption, 188
defining, 186
durations, 245
fill period
clip fills, 254
group default, 258
group fills, 256
group inheritance, 259
interactive timing, 187
priority classes
clips with no priority class, 193
defining, 188
effect on timing, 192
higher class interaction, 190
lower class interaction, 191
nesting, 193
peer interaction, 189
switching, 367
extension list, 485
external attribute, 295

F

fade effects, 329
fadeColor attribute, 334
file extension list, 485
fill attribute, 251
SMIL 1.0 and 2.0 differences, 142
transition value, 336
see also timing:fill period
fillDefault attribute, 258
film-to-video transfer, 72
fit attribute, 227
affect on clips, 227
clip scaling recommendations, 229
clip source tag, 220
filling the region, 229
illustration of effects, 228
interaction with registration points, 229
maintaining aspect ratio, 229
maintaining clip size, 229
overriding in clip source tag, 229
with RealText, 96
Flash
advancing scene to scene, 84

audio
export, 88
import, 79
bandwidth targets, 76
clip caching, 84
CPU use, 78
data spikes, 76
event sounds, 79
file size, 77
frame rate, 79
Get URL command
controlling RealONE Player, 83
pop-up windows, 306
sending URL to browser, 83
Go To command, 84
groups, 78
HTTP GET and POST commands, 87
key frames, 77
linear vs. non-linear, 75
Load Movie command
restrictions on, 85
SMIL in place of, 86
timeline behavior with, 86
mouse event trapping, 87
overview, 75
pausing
Flash clip, 83
RealONE Player, 84
Play command, 83
Player file export, 88
pop-up windows
examples, 308
links for, 306
Ram file with, 89
RealAudio issues
bandwidth division, 80
codec tips, 82
for 28.8 Kbps modems, 81
for 56 Kbps modems, 81
RealONE Player requirements, 76
RealServer requirements, 76
secure transactions, 87
seeking
through presentation, 83
time format for, 84
to Flash frame, 83
SMIL with, 89
starting
Flash clip, 83
RealONE Player, 84
stopping
Flash clip, 83
RealONE Player, 84
stream synchronization, 79
symbols, 78
timeline control, 82
tuning, 76
tweening, 79
Web server delivery, 428
focusInEvent value, 271
focusOutEvent value, 271
frame rates
Flash, 79
RealVideo, 59
video capture, 68
frequency response of RealAudio, 45
from attribute, 351
full-screen playback, 422

G

GIF, *see* images
graphics, *see* images
group tags, 173
groups
begin and end times, 241
durations, 245
relationship to timing, 237
see also exclusive groups
see also parallel groups
see also sequences

H

height attribute
<region/> tag, 208
<root-layout/> tag, 203
<topLayout> tag, 204
animating, 347, 349
clip source tag, 220
context window, 297
higher attribute, 190
horzRepeat attribute, 333
hot spots, *see* hyperlinks

href attribute, 290
HTML Help version of this guide, 5
HTML+Javascript version of this guide, 5
HTTP
compared to RTSP, 417
in presentation links, 417
in SMIL file, 153
see also Web server
hyperlinks
<a> tag, 283
activation methods, 282
actuate attribute, 293
alternate text, 293
animating, 350
<area/> tag, 283
automatic activation, 293
basic properties, 290
begin attribute, 284
coords attribute, 285
destinationLevel attribute, 306
end attribute, 284
external attribute, 295
fill period activity, 258
groups and, 282
hot spots
circular, 287
cropping, 289
image map programs, 289
mixing pixels with percentages, 289
overlapping, 284, 289
percentage decimal values, 289
polygonal, 288
rectangular, 286
sample coordinates, 290
scaling, 285
sizing, 285
tips, 289
href attribute, 290
key activation, 291
case-sensitivity, 291
indicating keys, 292
long description, 292
overlapping links, 292
usable keys, 291
long description, 283
media playback state
bandwidth issues, 300
controlling, 300
nesting, 282
nohref attribute, 291
overlapping, 283
recommendations, 279
regions and, 282
scaling in animated regions, 349
sendTo attribute, 296
shape attribute, 285
sound level adjustments, 306
sourceLevel attribute, 306
SourcePlaystate attribute, 300
streaming media
linking from Flash, 306
linking from RealText, 306
media playback state, 301
named media windows, 303
new media windows, 302
replacing a clip, 301
SMIL id link, 304
SMIL regions, 303
timeline offsets, 305
tabbing order, 294
target attribute, 299, 303
timed links, 284
URLs, 290
Web page
browser pop-up location, 300
context window, 297
see also context window
default browser, 297
frame targets, 299
HTML anchors, 300
main browser window, 296
named windows, 299
opening after a clip plays, 310
opening while a clip plays, 310
zoomlevel attribute, 307
see also clip source tags

I

id attributes
case-sensitivity, 138
clip source tags, 146
first characters, 139
length, 139

spaces in, 139
uniqueness, 138
images
caching, 154
durations, 245, 257
GIF or PNG transparency override, 162
in SMIL, 28
reliable transmission, 148
slow streaming example, 147
streaming speed, 146
supported formats, 28
see also RealPix
<img/> tag, 145
inBoundsEvent value, 270
inline switching, 365
interlaced video, 72
inverse-telecine filter, 72
iris wipes, 321
ISDN bandwidth targets, 32
ISPs and RealServer, 24

J
JavaScript with RealONE Player, 10
JPEG, *see* images

K
keystroke activation
for clip timing, 271
for hyperlinks, 291

L
LAN bandwidth use
lowering, 33
maximum, 32
language choices
codes, 487
setting, 367
laying out presentations
embedded playback, 412
with HTML, 413
with SMIL
in RealONE Player, 195
in Web page, 412
see also regions, 206
layout examples, 229
left attribute
<region/> tag, 208
<regPoint/> tag, 224
animating, 347, 349
clip source tag, 220
letterbox videos, 230
links
Ram file
to local files, 421
to RealServer, 421
to Web server, 421
Web page
to Web Server, 416
see also hyperlinks
local files, 25
logos, 432
long description, 170
longdesc attribute, 170
loss protection, 73
lower attribute, 191

M
manuals, where to find, 10
matrix wipes, 326
max attribute, 274
mediaOpacity attribute
animating, 349
using, 158
mediaRepeat attribute, 249
mediaSize attribute, 387
mediaTime attribute, 388
metafile, *see* Ram file
MIME types for Web servers, 418
min attribute, 274
modem bandwidth targets, 32
monitor size switching, 376
MPEG
converting to RealAudio, 54
converting to RealVideo, 70

N
namespaces
background on, 140
declaring but not using, 141
SMIL 2.0
customizations, 140
language profile, 134
support for in media players, 141

Netscape plug-in, *see* embedded playback
network connection speeds, 32
nohref attribute, 291
noise filters, 72
normal play time format
RealText, 98
SMIL, 240

O

open attribute, 204
OpenDML, 69
operating system switching, 374
outOfBoundsEvent value, 270
overdubbing preference, 369

P

<par> tag, 177
parallel groups
animations within, 343
authoring information, 178
bandwidth issues, 177
begin and end times, 241
defining, 177
durations, 245
fill period
clip fills, 254
group default, 258
group fills, 256
group inheritance, 259
transitions, 337
independent timelines, 179
normal end point, 178
region IDs for clips, 177
switching, 367
synchronizing clips
default synchronization values, 182
inheriting, 183
letting clips slip, 179
locking clip, 179
overview, 178
tolerance values, 185
defaults, 185
synchronizing groups, 181
nested group interactions, 183
with sequences, 174
see also timing
password authentication, 24
pauseDisplay attribute, 192
pay-per-view, 24
PDF version of this guide, 5
peers attribute, 189
planning a presentation, 23
plug-ins
see embedded playback
see RealONE Player
PNG, *see* images
pop-up windows
see regions:secondary media windows
<prefetch/> tag, 384
prefetching
bandwidth, 385
bits per second, 385
percentage of available, 386
bandwidth attribute, 385
CHTTP, 389
clipBegin attribute, 389
dangers, 384
data download size
bytes, 387
playing time, 388
recommended maximum, 387
effect on local playback, 383
examples, 390
mediaSize attribute, 387
mediaTime attribute, 388
overview, 383
RealAudio and RealVideo, 388
RealText, 389
SureStream, 388
synchronizing with a group, 384
testing, 390
timing attributes, 384
URLs, 384
base URL, 389
dynamic, 389
when to prefetch, 388
preroll, 32
presentation information overview, 165
priority classes, 188
paused clip display, 192
<priorityClass> tag, 188
protocols

CHTTP, 154
RTSP, 417
publishing tools, *see* RealProducer
push wipes, 329

Q

QuickTime
converting to RealAudio, 54
converting to RealVideo, 70

R

.ram extension, 425
Ram file
clip playback size, 422
comments, 423
creating manually, 420
linking
to local files, 421
to RealServer, 421
to Web server, 421
RealONE Player start mode, 422
replacing with Ramgen, 425
see also Ramgen
within a SMIL file, 149
Ramgen
options
altplay, 427
combining, 427
embed, 427
using, 425
`readIndex` attribute, 171
RealAudio
audio quality and bandwidth, 43
backward compatibility, 55
bandwidth characteristics, 44
codecs
frequency responses, 45
lossy nature, 43
mono music, 46
older, 48
sampling rates, 45
stereo music, 47
voice, 46
converting to other formats, 52
encoded information, 165
encoding instructions, 55
encoding tips, 54
Flash
audio export, 88
soundtrack, 79
input formats, 54
prefetching data, 388
RealVideo soundtracks, 57
sound quality, 43
streaming rates
multimedia, 55
standard, 44
streaming steps, 49
Web server playback, 428
with other clips, 44
see also audio
RealFlash, *see* Flash
RealGuide, 431
RealONE Player
autoupdate, 29
backward compatibility, 30
browser windows, 279
clip compatibility, 29
context window, 280
copyright protection
with RealServer, 30
with Web server, 428
doubling clip size, 422
download logo, 432
embedding in Web page
see embedded playback
full-screen playback, 422
JavaScript guide, 10
language choices
codes, 487
setting, 367
plug-in download, 29
RealVideo 8 codec support, 63
SDK, 10
SMIL source view, 141
theater mode, 422
toolbar mode, 422
VBScript guide, 10
RealPix
authoring guide, 10
bandwidth characteristics, 34
description, 29
through RealSlideshow, 28

Web server delivery, 429
RealProducer
overview, 27
SDK, 10
see also RealAudio
see also RealVideo
RealServer
administration guide, 10
administrator, 23
advanced features, 24
bandwidth constraints on, 24
SDK, 10
stream maximum, 23
stream thinning, 36
through ISPs, 24
RealSlideshow, 28
RealSystem
authoring kit, 10
SDK, 10
RealText
accented languages, 107
background color, 96
bandwidth characteristics, 93
bolding text, 115
broadcasting, 94
captions, 372
examples, 379
centering text, 114
character sets
big5, 108
default, 107
gb2312, 108
iso-2022-kr, 108
RealText version for, 100
iso-8859-1, 107
mac-roman, 108
coded characters, 119
RealText version for, 99
specifying, 106
us-ascii, 107
x-sjis, 108
Chinese character set, 108
coded characters, 118
mac-roman character set, 119
colors
hexadecimal values, 113
text, 112
text background, 112
window background, 112
comments, 93
crawl rate, 100
description, 29
duration
in SMIL presentation, 98
of clip, 97
troubleshooting, 98
erasing text, 104
examples, 119
generic window, 119
scrollingnews window, 121
teleprompter window, 122
tickertape window, 120
extra spaces, 102
features, 91
file
extension, 91
names, 93
structure, 92
fonts
Asian languages, 111
colors, 112
English and European languages, 109
text size, 111
freezing text, 104
guaranteed text delivery, 106
horizontal positioning, 105
horizontal rules, 115
horizontal text movement, 100
hyperlinks
color, 100
commands, 117
mail, 116
streaming presentation, 116
underlining, 100
Web page, 116
italicizing text, 115
Kanji character set, 108
Korean character set, 108
language support, 92
line breaks, 114
lists, 115

looping text, 101
Macintosh text entry, 108
overview, 91
paragraph tags, 114
pop-up windows
  examples, 308
  links for, 306
prefetching, 389
preformatted text, 114
RealONE Player command links
  pausing, 118
  playing, 118
  seeking, 117
scroll rate, 100
SMIL
  combining with other clips, 93
  fit attribute, 96
striking through text, 115
subtitles
  erasing each line, 104
syntax rules, 93
text size, 111
timing commands
  begin, 102
  end, 103
  with scrolling or crawling text, 104
transparency
  text backgrounds, 113
  window background, 96, 158
underlining text, 115
version numbers, 99
vertical positioning, 105
vertical text movement, 100
Web server delivery, 428
window size, 96
window types
  default attribute values, 95
  generic, 95
    example, 119
  marquee, 95
  scrollingnews, 95
    example, 121
  teleprompter, 95
    example, 122
  tickertape, 95
    colors, 105
    example, 120
    text positioning, 105
  word wrap, 101
RealVideo
  artifacts
    causes of, 60
    reducing, 72
  backward compatibility, 71
  bandwidth characteristics, 57
  codecs
    lossy nature, 59
    RealVideo 8, 63
    RealVideo G2, 64
    RealVideo G2 with SVT, 64
    standard, 64
  compressed input, 68
  converting to other formats, 67
  dimensions
    different sizes for different bit rates, 62
    recommended sizes, 61
    switching between, 378
  encoded information, 165
  encoding instructions, 70
  encoding tips, 70
  error correction, 73
  filters
    de-interlace, 72
    inverse-telecine, 72
    noise, 72
    resize, 72
  frame rates
    factors that affect, 59
    lowering for slow CPUs, 64
    overview, 59
    variable nature, 59
  input formats, 70
  prefetching data, 388
  production steps, 64
  quality guide, 61
  Scalable Video Technology, 64
  smooth motion option, 60
  soundtrack
    bandwidth ratio, 71
    RealAudio for, 57
  streaming rates
    modifying, 71

standard, 58
two-pass encoding, 73
variable bit rate encoding, 73
visual clarity
factors that affect, 61
overview, 60
RealProducer option for, 60
see also video
rebuffering, 32
<ref/> tag, 145
regAlign attribute, 221
region attribute in clip source tags, 213
regionName attribute
animating, 347
defining, 207
regions
assigning to clips, 213
audio clips, 213
audio volume control, 218
background colors, 215
changing in clip tag, 217
inheriting, 216
transparency
partial, 216
until clip plays, 216
bottom attribute, 208
clip scaling, 227
considerations for creating, 201
cropped at window boundaries, 213
defining, 206
examples
centering a video, 230
four offsets defined, 210
letterbox clip, 230
one offset defined, 212
overlapping regions, 212, 214
side-by-side clips, 231
size and two offsets defined, 210
two offsets defined, 211
width and height defined, 209
fit attribute, 227
height attribute, 208
id attribute, 207
layout tips, 199
left attribute, 208
multiple region playback, 233
multiple regions for one clip, 207
name attribute, 207
overlapping, 214
overview, 195, 196
percentage values, 208
positions, 208
resizing control, 206
reusing, 214
right attribute, 208
root-layout
defining, 203
overview, 195
sizing
considerations, 200
double-screen mode, 200
example, 200
full-screen mode, 200
RealONE Player controls, 200
secondary media windows, 197
close attribute, 204
considerations for using, 201
defining, 204
example, 232
hyperlinked window comparison, 198
open attribute, 204
pop-up location, 205
viewer interaction, 205
sizes
pixels and percentages, 208
decimal percentages, 213
mixing, 213
resize behavior, 213
setting, 208
soundLevel attribute, 218
stacking order, 214
subregions
attribute inheritance, 219
background colors, 219
considerations for creating, 201
defining, 218
nesting, 219
overview, 196
registration point comparison, 202
single-use, 220
z-index attribute, 219

tag summary, 202
top attribute, 208
transparency, 215
width attribute, 208
z-index attribute, 214
registration points
alignment values, 221
clip source tags, 222
common values
in clip source tags, 223
in <regPoint/> tags, 225
considerations for creating, 202
default positioning, 226
defining in layout, 224
fit attribute interaction, 227
ID values, 226
methods of creating, 221
misalignment problems, 222, 226
overview, 199
pixels and percentages
defining, 224
mixing, 226
recommendations, 226
positioning, 224
relationship to regions, 226
reusing in clips, 226
subregion comparison, 202
regPoint attribute, 222, 224
<regPoint/> tag, 224
see also registration points
relative links
in Ram file, 420
in SMIL, 152
reliable transmission, 148
repeat(n) value, 268
repeatCount attribute, 247
repeatDur attribute, 247
repeatEvent value, 268
repeating clips, *see* timing:repeating clips
resize filters, 72
resizeBehavior attribute, 206
restart attribute, 274
restartDefault attribute, 275
right attribute, 208
<region/> tag, 208
<regPoint/> tag, 224
animating, 347, 349
clip source tag, 220
rn: prefix, 139
rollover events, 269
<root-layout/> tag, 203
see also regions
.rpm extension, 425
RTSP
compared to HTTP, 417
in presentation links, 417
in SMIL file, 153
overview, 417
port number, 153

S

sample files, 5
sampling rates, 45
Scalable Video Technology (SVT), 64
scaling clips in regions, 227
secondary media windows, *see* regions:secondary media windows
secure transactions with Flash, 87
sendTo attribute
namespace declaration, 296
on Linux, 296
on Macintosh, 296
with <a> tags, 296
<seq> tag, 175
sequences
animations within, 343
authoring information, 176
begin and end times, 241
defining, 175
durations, 245
fill period
clip fills, 253
group default, 258
group fills, 256
group inheritance, 259
transitions, 336
Next Clip command, 175
seeking through, 175
single presentation vs. multiple clips, 175
switching, 367
with parallel groups, 174

<set/> tag
compared to <animate/>, 342
using, 360
shape attribute, 285
shielded cables, 51
Shockwave Flash, *see* Flash
show attribute, 301
slide wipes, 329
SMIL 1.0
RealPlayer support, 129
updating to SMIL 2.0, 143
SMIL 2.0
advantages of, 127
attribute changes from SMIL 1.0, 143
attribute format, 135
bandwidth characteristics, 34
binary tags, 137
body section, 134
camel case values, 136
case-sensitivity, 136
closing tag, 134
coded characters, 167
comments, 137
customized attributes, 139
extensions, 139
functional changes from SMIL 1.0, 142
general rules, 133
header, 134
coded characters, 167
hyphenated attributes, 136
id attributes, 138
indentation, 138
language codes, 487
layout, *see* regions
links, *see* hyperlinks
major features, 127
media player interoperability, 132
modules, 129
namespace, 134
overview, 127
prefixes, 139
presentation information, 169
profiles, 131
proprietary datatypes with, 132
quotation marks for values, 136
RealONE Player compliance, 131
.smil extension, 133
SMIL file as a clip, 150
source file viewing, 141
specification, 127
syntax errors, 137
tag format, 135
text display, 29
timing, *see* timing
title, author, copyright, 169
unary tags, 137
viewing SMIL source, 141
with Web server playback, 429
SMIL animation
accumulate attribute, 356
additive attribute, 355
attributeName attribute, 346
audio levels, 348
begin times, 344
by attribute, 351
calcMode attribute, 353
clip stacking order, 348
clips, 349
colors, 358
discrete values, 353
durations, 344
examples, 341
flowing from point to point, 354
freezing, 344
from attribute, 351
hyperlinks
changing coords values, 350
scaling in animated regions, 349
in clip tags, 343
in parallel groups, 343
in sequences, 343
incremental, 355
jumping from point to point, 353
linear values, 354
motion, 359
multiple animation points, 352
overview, 341
paced values, 354
regions, 347
repeating, 345
repeating and growing, 356

root-layout, 346
selecting attribute to animate, 346
setting values instantly, 360
simultaneous, 345
start and stop values, 350
tags, 342
targetElement attribute, 346
time manipulations, 360
timing attributes, 344
to attribute, 351
transparency, 349
values attribute, 352
.smil extension, 133
SMPTE code, 318
SND conversion to RealAudio, 54
software development kits (SDKs), 10
soundLevel attribute
animating, 348
using, 218
sourceLevel attribute, 306
sourcePlaystate attribute, 300, 301
special effects
see SMIL animation
see transition effects
stream thinning, 36
streaming
speeds for network connections, 32
versus downloading, 417
Web server, 416
subregions, *see* regions
subtitling preference, 369
SureStream
backward compatibility, 30
downshifting, 35
overview, 35
RealAudio codecs, 45
switching, 371
s-video, 68
<switch> tag, 363
switching
audio descriptions, 372
bandwidth, 370
captions, 372
example, 379
color depth, 376
CPU type, 373
default option
adding, 364
filler clip, 367
when not to use, 367
examples
captions, 379
different video dimensions, 378
multiple attributes, 377
group switching, 367
hyperlink inclusion, 367
inline, 365
language choices, 367
codes, 368
layouts, 367
monitor size, 376
multiple attributes
example, 377
nested, 367
example, 378
operating system, 374
overview, 36, 363
subtitles or overdubbing, 369
test attributes
list of, 366
multiple attributes, 367
versus interactive choices, 363
syncBehavior attribute, 179
syncBehaviorDefault attribute, 182
synchronizing clips in parallel groups, 178
syncTolerance attribute, 185
syncToleranceDefault attribute, 185
systemAudioDesc attribute, 372
systemBitrate attribute, 370
systemCaptions attribute, 372
systemCPU attribute, 373
systemLanguage attribute, 367
systemOperating system attribute, 374
systemOverdubOrSubtitle attribute, 369
systemScreenDepth attribute, 376
systemScreenSize attribute, 376

T

tabindex attribute, 294

target attribute, 299, 303
targetElement attribute, 346
technical support, 10
testing presentations, 430
text, 29
text streaming, *see* RealText
<text/> tag, 146
<textstream/> tag, 146
theater mode for RealONE Player, 422
time manipulations, 360
Timecast, *see* RealGuide
timing
  advanced, 261
    element repeat, 268
    element start or stop, 266
    interactive events, 262
    keyboard events, 271
    media markers, 274
    min and max values, 274
    mouse event, 269
    negative offset values, 265
    positive offset values, 263
    scheduled events, 262
    secondary window events, 273
    syntax, 261
    wallclock timing, 274
  animated GIF modification, 249
  basic, 237
  begin attribute, 240
  clipBegin attribute, 242
  clipEnd attribute, 242
  delaying clip playback, 240
  dur attribute, 243
    compared to end, 243
  durations, 243
    groups, 245
    images, 245
    indefinite, 244
    normal clip length, 244
  end attribute, 241
    compared to dur, 243
  exclusive groups, 187
    begin and end times, 241
    durations, 245
  fill attribute, 251
fill period, 251
  automatic fill, 252
  default fills, 258
  exclusive group clips, 254
  groups, 256
  indefinitely visible clip, 255
  parallel group clips, 254
  sequential clips, 253
  SMIL 1.0 and 2.0 differences, 142
  summary, 255
keyboard events
  case-sensitivity, 272
  indicating keys, 272
  long description, 272
  usable keys, 272
mediaRepeat attribute, 249
multiple time values, 266
overview, 237
parallel groups
  begin and end times, 241
  durations, 245
  group endpoint
    first clip, 246
    last clip, 246
    specific clip, 246
partial clip play, 251
relationship to groups, 237
repeatCount attribute, 247
repeatDur attribute, 247
repeating clips
  bandwidth management, 250
  indefinite number of times, 248
  repeating cycle length, 248
  server streams used, 250
  specific amount of time, 247
  specific number of times, 247
  total playing time, 248
restarting elements, 274
  group defaults, 275
sequences
  begin and end times, 241
  durations, 245
tenths of seconds display, 239
values
  normal play time format, 240
  shorthand, 239

Web server delivery issues, 429
timing a presentation
internal timelines, 37
timeline management, 39
timeline synchronization, 37
variable timelines, 38
with multiple clips, 38
title attribute, 168
to attribute, 351
toolbar mode for RealONE Player, 422
top attribute
<region/> tag, 208
<regPoint/> tag, 224
animating, 348, 350
clip source tag, 220
<topLayout/> tag, 204
see also regions
topLayoutCloseEvent value, 273
topLayoutOpenEvent value, 273
transIn attribute, 335
transition effects
animation comparison, 316
assigning to clips, 335
audio impact, 316
borderColor attribute, 334
borders
blends, 334
colors, 334
pixel width, 334
borderWidth attribute, 334
clock wipes, 323
direction attribute, 331
dur attribute, 331
duration
changing, 331
default, 316
edge wipes, 318
endProgress attribute, 332
examples
color fade, 338
crossfade, 339
fadeColor attribute, 334
fades, 329
colors, 334
fill attribute
parallel groups, 337
sequences, 336
horzRepeat attribute, 333
id attribute, 318
instantaneous effects, 333
iris wipes, 321
layout considerations, 316
matrix wipes, 326
multiple clips, 317
overview, 315
partial effects
defining, 332
fill attribute, 333
push wipes, 329
repeating, 333
slide wipes, 329
SMPTE code, 318
startProgress attribute, 332
subtype attribute, 318
tag summary, 317
timeline impact, 316
transIn attribute, 335
transOut attribute, 335
type attribute, 318
vertRepeat attribute, 333
<transition/> tag, 317
see also transition effects
transOut attribute, 335
transparency
clips
color for transparency, 162
for a specific opaque color, 159
for all opaque colors, 158
for background color, 158
supported clip types, 217
region backgrounds
full, 215
partial, 216
two-pass encoding, 73

V

values attribute, 352
variable bit rate encoding, 73
VBScript with RealONE Player, 10
vertRepeat attribute, 333
VHS format, 66

video
see also RealVideo
capture
cards, 26
disk space, 69
file size limit, 69
formats, 68
frame rates, 68
requirements, 69
screen size, 68
editing programs, 26
interlaced, 72
lighting, 67
minimizing movement, 67
motion resolution, 67
production tools, 26
recording tips, 66
source formats, 66
staging shots, 66
streaming steps, 64
s-video, 68
24-bit depth, 68
Video for Windows, 68
<video/> tag, 146
visual quality of RealVideo, 60

W wallclocks for broadcasts, 274
WAV conversion to RealAudio, 54
Web page playback, 395
Web server
MIME type configuration, 418
playback
instructions, 416
limitations, 428
unsecure clips, 428
widescreen video display, 230
width attribute
<region/> tag, 208
<root-layout/> tag, 203
<topLayout> tag, 204
animating, 347, 348, 350
clip source tag, 220
context window, 297
<window> tag, 92
wipes, *see* transition effects
World Wide Web Consortium (W3C), 127

X XML namespace, 134
xmlns attribute, 134

Z z-index attribute, 214
animating, 348, 350
clip source tag, 220
default value, 215
duplicate values, 215
negative integers, 215
recommended values, 215
<root-layout/> tag, 215
subregions, 219
zoomlevel attribute, 307